INPUT
2
L. 2800
CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER
DE AGOSTINI

CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER

Pubblicazione a fascicoli settimanali
edita dall'Istituto Geografico De Agostini

volume I - fascicolo 2

Direttori: Achille Boroli - Adolfo Boroli

Direzione editoriale: Mario Nilo; *settore fascicoli:* Jason Vella

Redazione dell'edizione italiana a cura della:
Logical Studio Communication
Traduzione dall'inglese a cura di: Daniel Quinn

Coordinamento grafico: Otello Geddo
Coordinamento fotografico a cura del Centro Iconografico dell'Istituto Geografico De Agostini
Direzione: Novara (28100), via Giovanni da Verrazano 15 - tel. (0321) 471201-5
Redazione: Milano (20149), via Mosè Bianchi 6 - tel. (02) 4694451

Programma di abbonamento. Condizioni di abbonamento all'intera opera in 52 fascicoli, completa di copertine e di risguardi per la confezione dei 6 volumi dell'opera:
a) in un unico versamento anticipato di L. 180 000 in Italia, L. 225 000 all'estero;
b) in 4 versamenti trimestrali consecutivi e anticipati di L. 45 250 ciascuno.
La forma di abbonamento *b* è ammessa soltanto in Italia.
Agli abbonati all'intera opera sono riservato in dono "2 cassette di videogiochi" oppure, in alternativa, "5 cassette da registrare" (Aut. Min. conc.).
I versamenti possono essere effettuati a mezzo assegno bancario oppure sul c/c postale n. 111286 intestato all'Istituto Geografico De Agostini - Novara.
Amministrazione, abbonamenti e servizio arretrati: Istituto Geografico De Agostini - Novara (28100), via Giovanni da Verrazano 15 - tel. (0321) 471201-5.
Copertine e risguardi per i volumi dell'opera saranno messi in vendita a L. 6000 la copia (L. 7500 all'estero).
Le copie arretrate saranno disponibili per un anno dal completamento dell'opera e potranno essere prenotate nelle edicole o direttamente presso l'Editore. Per i fascicoli arretrati, trascorse 12 settimane dalla loro pubblicazione, è applicato un sovrapprezzo di L. 400 sul prezzo di copertina in vigore al momento dell'evasione dell'ordine. Spedizione contro rimessa di pagamento anticipato; non vengono effettuate spedizioni contrassegno.
L'Editore si riserva la facoltà di modificare il prezzo nel corso della pubblicazione, se costretto da mutate condizioni di mercato.

Registrato presso il Tribunale di Novara n. 11 in data 19-5-1984.
Direttore responsabile: Emilio Bucciotti

Spedizione in abbonamento postale Gruppo II/70 (Autorizzazione della Direzione provinciale delle PP.TT. di Novara).
Distribuzione A. & G. Marco - Milano, via Fortezza 27 - tel. (02) 2526.
Pubblicazione a fascicoli settimanali. Esce il martedì.
Stampato in Italia - I.G.D.A. Officine Grafiche, Novara - 308410.

Referenze dei disegni e delle fotografie:
Copertina: Pete Seaward. Pagg. 34, 37 Chen Ling. Pagg. 38, 45 Phil Dobson. Pagg. 46-53 Pete Seaward. Pagg. 54-59 Dick Ward. Pagg. 60-63 Andrew MacConville.

INPUT È STUDIATA APPOSITAMENTE PER:

Lo SPECTRUM della Sinclair (versioni 16K e 48K), il COMMODORE 64, l'ELECTRON ed il BBC della ACORN, il DRAGON 32.
Comunque, molti dei programmi e dei testi sono adatti anche per: lo ZX81 della SINCLAIR, il COMMODORE VIC 20 ed il TANDY COLOUR COMPUTER con 32K ed il BASIC esteso.

I seguenti simboli identificano i programmi o le spiegazioni adatte a ciascun computer:

 SPECTRUM

 COMMODORE 64

 ELECTRON e BBC

 DRAGON 32

 ZX81

 VIC 20

 TANDY TRS80 COLOUR COMPUTER

IL COMPUTER PRENDE DECISIONI

- QUALE STRADA PRENDERE
- IL CALCOLO DELLE MEDIA
- DECISIONI PIÙ COMPLESSE
- UNA SEMPLICE FRUIT MACHINE
- L'USO DI **IF ... THEN ... ELSE**

Per quanto 'senza cervello', il computer può prendere decisioni logiche... se il programma è giusto. Ecco come usare IF ... THEN per fargli fare delle scelte

Una delle doti che differenziano un computer da una calcolatrice è la capacità di compiere delle scelte. Questa utile caratteristica permette ai programmi di ramificarsi in differenti direzioni per eseguire particolari gruppi di istruzioni, a seconda del risultato di un confronto.

Uno dei metodi che rende ciò possibile prevede l'uso della struttura IF ... THEN, dal significato piuttosto intuitivo: SE (IF) la condizione è vera, ALLORA (THEN) esegui una certa operazione. Un esempio di questa struttura è già stato dato a pagina 3. Eccone un altro:

IF A < 18 (ossia se A è minore di 18) THEN PRINT "minorenne"

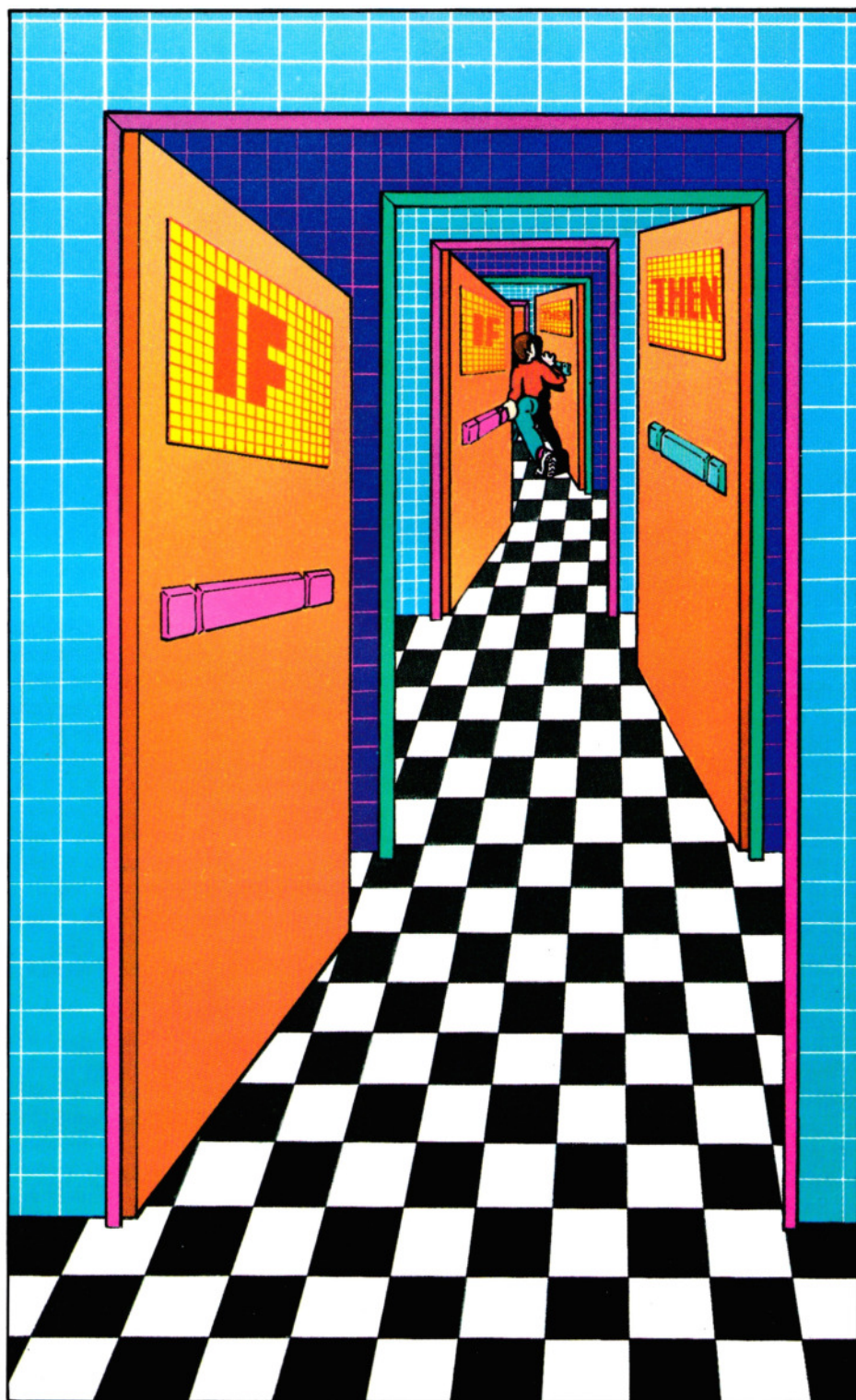
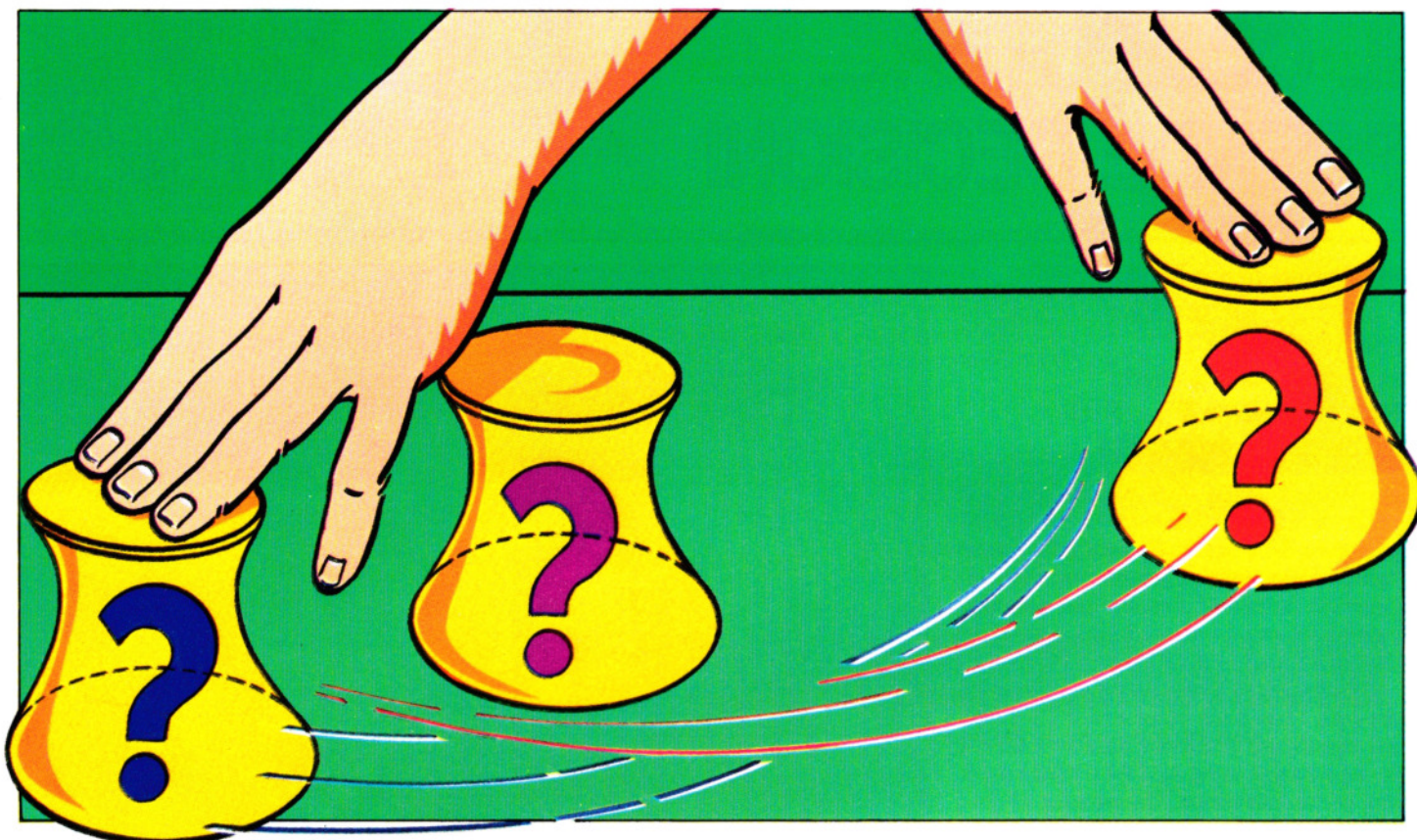
Quando il computer incontra la parola IF, controlla se la successiva condizione è vera. Se lo è, il programma procede con l'esecuzione delle istruzioni che seguono la parola THEN. Se, invece, la condizione non è vera, l'esecuzione passa direttamente alla linea di programma successiva.

IL CALCOLO DELLA MEDIA

Vediamo come si calcola la media di una serie di valori:

Sullo ZX81 si ometta lo STOP alla linea 40 e si aggiunga:

```
45 IF N= -99 THEN GOTO 80
80STOP
```

```
10 PRINT "IMMETTERE I VALORI"
20 PRINT "(-99 PER TERMINARE)"
25 LET T=0
26 LET C=0
30 INPUT N
40 IF N= -99 THEN PRINT "MEDIA=";T/C:
   STOP
50 LET T=T+N
60 LET C=C+1
70 GOTO 30
```

```
10 PRINT "IMMETTERE I VALORI"
20 PRINT "(−99 PER TERMINARE)"
30 INPUT N
40 IF N=−99 THEN PRINT "MEDIA=";T/C:
   END
50 LET T=T+N
60 LET C=C+1
70 GOTO 30
```

Le istruzioni dicono di immettere una lista di valori e di usare −99 (tra breve vedremo perché) per terminare la lista. Le linee 25 e 26 (necessarie soltanto sullo Spectrum) assegnano il valore zero al totale T e al contatore C. La linea 30 prende il numero immesso, la linea 50 lo somma al totale parziale e la linea 60 tiene il conto dei valori inseriti, sommando 1 al valore di C ad ogni immissione. Finché i valori inseriti sono diversi da −99, il computer ignora la linea 40 e, attraverso la 70, torna alla 30 per l'input di un nuovo valore. Se questo è −99, la condizione alla linea 40 è vera, provocando la visualizzazione della media (ossia di T/C, totale diviso il numero di valori) e il programma termina.

In questo caso, −99 è un *numero limite* o *di comodo*, utile al controllo del programma.

TRIPLICE SCELTA

Che fare se si vuole decidere tra tre o più percorsi diversi? La cosa è facile quanto scegliere tra due sole alternative, come si può vedere dalla seguente rielaborazione dell'indovinello presentato nella precedente lezione:

```
5 CLS
10 LET N=RND(20)
20 PRINT "HO APPENA PENSATO UN
   NUMERO"
30 PRINT "...RIESCI AD INDOVINARE QUALE
   SIA?"
40 INPUT G
50 IF G=N THEN PRINT "GIUSTO, BEN
   FATTO!":FOR D=1 TO 2000:NEXT D:
   GOTO 10
60 IF G<N THEN PRINT "TROPPO BASSO,
   PROVA ANCORA!"
70 IF G>N THEN PRINT "TROPPO ALTO,
   PROVA ANCORA!"
80 GOTO 40
```

```
5 CLS
10 LET N=INT (RND*20)+1
20 PRINT "HO APPENA PENSATO UN
   NUMERO"
30 PRINT "...RIESCI AD INDOVINARE QUALE
   SIA?"
40 INPUT G
50 IF G=N THEN PRINT "GIUSTO, BEN
   FATTO!": PAUSE 100: GOTO 10
60 IF G<N THEN PRINT "TROPPO BASSO,
   PROVA ANCORA!"
70 IF G>N THEN PRINT "TROPPO ALTO,
   PROVA ANCORA!"
80 GOTO 40
```

```
5 PRINT "♥"
10 LET N=INT(RND(1)*20+1)
20 PRINT "HO APPENA PENSATO UN
   NUMERO"
30 PRINT "...RIESCI AD INDOVINARE QUALE
   SIA?"
40 INPUT G
50 IF G=N THEN PRINT "GIUSTO, BEN
   FATTO!":FOR D=1 TO 1000:NEXT D:
   GOTO 5
60 IF G<N THEN PRINT "TROPPO BASSO,
   PROVA ANCORA!"
70 IF G>N THEN PRINT "TROPPO ALTO,
   PROVA ANCORA!"
80 GOTO 40
```

Alla linea 10 viene scelto un numero a caso tra 1 e 20, poi le linee da 20 a 40 chiedono d'indovinarlo. A ogni tentativo, il com-

puter passa in rassegna le linee 50, 60 e 70, per controllare quale condizione sia vera. Supponiamo, ad esempio, che il tentativo sia troppo basso. In tal caso, il computer trova che la condizione alla linea 50, (G = N) è falsa e passa alla linea 60. Qui la condizione è vera: G è minore di N. Pertanto, visualizza il messaggio "TROPPO BASSO, PROVA ANCORA!". L'esecuzione, necessariamente, passa alla linea 70, ma, trovando che la condizione è falsa, passa direttamente alla linea 80, che rimanda a un nuovo tentativo.

E se il tentativo era, invece, troppo alto o era quello giusto? Osservando il programma la risposta è semplice.

Questo programma funziona benissimo, ma ha un inconveniente: continua a chiedere di indovinare numeri, sia che si voglia giocare o no. Una soluzione migliore è quella di far chiedere se si desidera un altro tentativo. Le poche linee che seguono servono proprio a questo. Viene di nuovo impiegato IF ... THEN, ma in questo caso il confronto è su delle lettere e non su dei numeri.

Microtip

Problemi con gli operatori?

A chi non è familiare con i simboli 'maggiore' o 'minore' (chiamati *operatori*), questi segni possono sembrare strani. Essi vanno pensati come dei cunei '< >'. In >, la parte sinistra è *maggiore* dell'estremità a punta. In < è vero il contrario. Cosicché, A > B si legge 'A è maggiore di B'. Se aggiungiamo anche =, significa che A può anche essere uguale a B.

Ecco un elenco completo delle varie combinazioni:

A=B : A uguale a B
A>B : A maggiore di B
A<B : A minore di B
A>=B : A maggiore o uguale a B
A<=B : A minore o uguale a B
A<>B : A diverso da B

Sugli Acorn, occorre scrivere gli operatori secondo un preciso ordine: se scriviamo, ad esempio, A=<B, in BASIC questo è considerato errore.

Sullo Spectrum, gli operatori complessi (quale <=, ad esempio), devono essere immessi mediante l'apposito tasto. Digitando < seguito da = l'intera linea non verrebbe accettata.

```
100 PRINT "VUOI CONTINUARE ANCORA
    (S/N)?"
110 LET A$=GET$
120 IF A$="S" THEN RUN
130 END
```

```
100 PRINT "VUOI CONTINUARE ANCORA
    (S/N)?"
110 LET A$=INKEY$ : IF A$="" THEN
    GOTO 110
120 IF A$="S" THEN RUN
```

```
100 PRINT "VUOI CONTINUARE ANCORA
    (S/N)?"
110 GET A$: IF A$="" THEN 110
120 IF A$="S" THEN RUN
130 END
```

Desiderando aggiungere queste linee, occorre cambiare la linea 50 del precedente programma in:

```
50 IF G=N THEN PRINT "GIUSTO, BEN
   FATTO!": GOTO 100
```

La linea 110 aspetta che si prema un tasto. Premendo una S (maiuscola), il programma riparte da capo, ma qualsiasi altro tasto (compresa la s minuscola!) termina l'esecuzione.

Queste linee possono essere facilmente aggiunte a qualsiasi programma di gioco, per offrire una rapida via d'uscita.

DOPPIO CONTROLLO

A volte occorre che il computer verifichi se due o più condizioni sono contemporaneamente vere, prima di proseguire. Un modo per far questo è impiegare speciali parole chiave o simboli detti *operatori*.

Osserviamo questa linea di programma:

```
100 IF D$="DOMENICA" AND T=1500
    THEN PRINT "ATTENTO: C'È LA
    PARTITA!"
```

Quando si usa la parola chiave AND tra due condizioni, allora le condizioni devono essere entrambe vere, perché vengano eseguite le istruzioni che seguono THEN, altrimenti l'esecuzione passa alla linea successiva. In questo esempio, soltanto se è domenica e sono le tre pomeridiane viene visualizzato il messaggio.

Un altro esempio è:

```
200 IF P$="BRODO" OR P$="TAPIOCA"
    THEN PRINT "OGGI NON MI SENTO
    AFFAMATO!"
```

Questa linea usa la parola chiave OR e il messaggio appare, purché almeno una delle due condizioni sia vera. Il test può diventare alquanto complesso, se ci sono molte condizioni da verificare e se vengono usate AND e OR assieme. In tal caso, è bene usare delle parentesi per assegnare un ordine di precedenza. Per esempio, una linea in un gioco d'avventura potrebbe somigliare a:

```
2000 IF P=14 AND (C$="SPADA" OR C$
     ="PUGNALE") THEN PRINT "HAI UCCISO
     IL MOSTRO"
```

Questa condizione è vera (e si riesce a sopraffare l'avversario) soltanto se si è nella posizione 14 e (AND) si sta impugnando una spada o (OR) un pugnale. Ma proviamo a cambiare le parentesi:

```
2000 IF (P=14 AND C$="SPADA") OR C$
     ="PUGNALE" THEN PRINT "HAI UCCISO
     IL MOSTRO"
```

Adesso, la condizione è vera se si è nella posizione 14 e (AND) si è armati di una spada oppure (OR) se si è in un qualsiasi luogo, ma armati di un pugnale, il che non è affatto la stessa cosa! Le parentesi sono essenziali per comunicare al computer, con esattezza, la priorità delle condizioni.

Per guadagnare spazio, si possono combinare più strutture IF ... THEN in una sola linea?

Generalmente, non è affatto consigliabile. Il funzionamento delle IF ... THEN è tale, in BASIC, che se la prima condizione è vera, il resto della linea, dopo THEN, viene eseguito, altrimenti no. Così, in questa linea:

```
70 IF X=Y THEN PRINT "FUORI TEMPO":
   LET conto=conto -1: GOTO 30
```

non viene eseguita alcuna istruzione che si trovi dopo la Y, salvo nel caso in cui X=Y. Talvolta, comunque, può essere utile affiancare più IF ... THEN. In queste linee:

```
70 IF X=Y THEN PRINT "FUORI TEMPO":
   IF conto >0 then LET conto = -1
80 IF X=X AND conto= 0 THEN PRINT
   "Fine del gioco"
```

il giocatore si ritrova con 0 punti soltanto se ne aveva solo 1, ma, a causa della struttura della linea 70, il messaggio "FUORI TEMPO" è visualizzato in tutti i casi.

UNA SEMPLICE FRUIT MACHINE

Ecco un programma di gioco, che fa buon uso di AND ed OR. Proviamo a giocare:

```
20 LET M=50
30 CLS
40 LET M=M-5
50 IF M<0 THEN PRINT "PECCATO, HAI
   PERSO TUTTO!": STOP
60 LET A=INT (RND*12)+130
70 LET B=INT (RND*12)+130
80 LET C=INT (RND*12)+130
210 PRINT PAPER 0; INK 4;AT 10,14;CHR$
   A;AT 10,16;CHR$ B;AT 10,18;CHR$ C
220 IF A=B AND B=C THEN PRINT AT
   13,2;"HAI FATTO UN JACKPOT... DI 50$":
   LET M=M+50
230 IF (A=B OR B=C) AND A<>C THEN
   PRINT AT 13,9;"HAI VINTO 10$": LET M
   =M+10
240 PAUSE 25
250 PRINT AT 15,8;"UN ALTRO GIRO (s/n)?":
   PRINT TAB 10;"HAI ANCORA";M;"$"
260 IF INKEY$="" THEN GOTO 260
270 IF INKEY$="n" THEN STOP
280 GOTO 30
```

Che fare se si ottengono dei messaggi d'errore, eseguendo un RUN di un programma appena trascritto?

Può darsi che ci sia veramente un errore nel programma, ma spesso si tratta di qualche sbaglio nella trascrizione. Ecco i più comuni:

- Confondere la I maiuscola o la l minuscola con il numero 1
- Confondere la O maiuscola con il numero 0
- Dimenticare gli apici al termine di un'istruzione PRINT
- In una frase DATA, omettere la virgola di separazione tra due valori (il numero dei dati non corrisponde più)
- Omettere un segno 'meno' (nei programmi grafici ciò potrebbe 'suggerire' al computer di visualizzare qualcosa 'fuori dallo schermo').
- Omettere il punto e virgola al termine di una linea (provocando confusione nella disposizione dei dati sullo schermo)

```
20 LET M=50
30 CLS
40 LET M=M-5
50 IF M<0 THEN PRINT "PECCATO, HAI
   PERSO TUTTO!":END
60 LET A=RND(12)+192
70 LET B=RND(12)+192
80 LET C=RND(12)+192
210 PRINT@ 237,CHR$(A):PRINT@
   239,CHR$(B):PRINT@ 241,CHR$(C)
220 IF A=B AND B=C THEN PRINT@258,
   "HAI FATTO UN JACKPOT... DI 50$": LET
   M=M+50
230 IF (A=B OR B=C) AND A<>C THEN
   PRINT@265,"HAI VINTO 10$":LET M=M
   +10
240 FOR D=1 TO 500:NEXT
250 PRINT@ 327,"UN ALTRO GIRO (S/N)?":
   PRINT@ 361,"HAI ANCORA";M;"$"
260 LET K$=INKEY$:IF K$="" THEN
   GOTO 260
270 IF K$="S" THEN GOTO 30
280 END
```

```
20 LET M=50
30 CLS
40 LET M=M-5
50 IF M<0 THEN PRINT "PECCATO, HAI
   PERSO TUTTO!":END
60 LET A=RND(12)+224
70 LET B=RND(12)+224
80 LET C=RND(12)+224
210 PRINT TAB(17,10);CHR$147;CHR$A;"□";
   CHR$B;"□";CHR$C
220 IF A=B AND B=C THEN PRINT
   TAB(7,12) "HAI FATTO UN JACKPOT...DI
   50$": LET M=M+50
230 IF (A=B OR B=C) AND A<>C THEN
   PRINT TAB(14,12) "HAI VINTO 10$": LET
   M=M+10
240 FOR D=1 TO 1500: NEXT
250 PRINT TAB(13,16) "UN ALTRO GIRO
   (S/N)?": PRINT TAB(15,17) "HAI
   ANCORA";M;"$"
260 LET K$=GET$
270 IF K$="S" THEN GOTO 30
280 END
```

Per il Vic, cambiare la linea 10 in 40 POKE 36879,8. Cambiare TAB (15) nella linea 210 in TAB (3). Omettere TAB (5) nella linea 220 e cambiare infine TAB (13) nella linea 230 in TAB (4).

```
10 POKE 53280,0: POKE 53281,0:PRINT
   CHR$(30)
20 LET M=50
30 PRINT "♥"
40 LET M=M-5
```

L'uso di REPEAT ... UNTIL

I computer Acorn possiedono la struttura REPEAT ... UNTIL, che può spesso rimpiazzare la IF ... THEN ... GOTO. Ciò è particolarmente utile quando un certo gruppo di istruzioni deve essere ripetuto più volte, fino a quando non si verifica una specifica condizione.

In un programma di gioco, ad esempio, la condizione può essere l'esaurimento delle munizioni:

```
50 (inizio del programma principale)
200 IF bombe=0 THEN PRINT "Fine del
   gioco!":END
210 GOTO 50
```

Ma con REPEAT ... UNITL, ciò diventa:

```
45 REPEAT
50 (inizio del programma principale)
200 UNTIL bombe=0
210 PRINT "Fine del gioco!":END
```

Le due versioni sono equivalenti, ma la seconda è più chiara e semplice, oltre a essere di più rapida esecuzione.

```
50 IF M<0 THEN PRINT "PECCATO, HAI
   PERSO TUTTO!": END
60 LET A=INT(RND(1)*4)+1
70 LET B=INT(RND(1)*4)+1
80 LET C=INT(RND(1)*4)+1
90 IF A=1 THEN LET A=97
100 IF A=2 THEN LET A=115
110 IF A=3 THEN LET A=120
120 IF A=4 THEN LET A=122
130 IF B=1 THEN LET B=97
140 IF B=2 THEN LET B=115
150 IF B=3 THEN LET B=120
160 IF B=4 THEN LET B=122
170 IF C=1 THEN LET C=97
180 IF C=2 THEN LET C=115
190 IF C=3 THEN LET C=120
200 IF C=4 THEN LET C=122
210 PRINT "♥▣▣▣▣▣▣▣▣▣
   ▣"TAB(15)CHR$(A)SPC(3)CHR$(B)SPC(3)
   CHR$(C)
220 IF A=B AND B=C THEN PRINT TAB(5)
   "▣▣▣▣HAI FATTO UN JACKPOT...DI
   50$":LET M=M+50
230 IF (A=B OR B=C) AND A<>C THEN
   PRINT TAB(13) "▣▣▣▣HAI VINTO
```

```
   10$":LET M=M+10
240 FOR D=1TO1500:NEXT
250 PRINT "▓▓▓▓UN ALTRO GIRO
   (S/N)?...HAI ANCORA";M;"$"
260 GET K$:IF K$<>"S" AND K$<>"N"
   THEN GOTO 260
270 IF K$="S" THEN GOTO 30
280 END
```

In questo programma vengono usate diverse IF ... THEN. La prima si trova alla linea 50, e controlla semplicemente che si abbia denaro sufficiente per giocare. In caso positivo, il resto della linea è ignorato, altrimenti appare un messaggio e il gioco finisce.

Le linee da 60 a 80, scelgono tre numeri a caso, mentre la linea 210 li converte in caratteri, visualizzandoli al centro dello schermo.

Sul Sinclair, sul Dragon e sugli Acorn la conversione è molto semplice, ma sul Commodore occorrono dodici linee supplementari per convertire i numeri casuali da 1 a 4 nel codice di uno dei semi delle carte (quadri, cuori, fiori e picche), presente come carattere grafico nella ROM del computer. La linea 210 provvede alla visualizzazione, come per le altre macchine. Alla linea 220, se i tre caratteri sono identici, si vince il 'Jackpot' ed il nostro denaro aumenta di 50 dollari.

Alla linea 230 si vincono 10 dollari, se due dei caratteri sono uguali (A=B o B=C), purché non siano quelli esterni. Se non si possiede una combinazione vincente, le linee 220 e 230 vengono ignorate e, dopo una breve pausa provocata da un ciclo, si passa a una routine che chiede se si desidera un'altra partita.

IF ... THEN ... ELSE

Su alcuni computer (non lo Spectrum né il Commodore, però), si può usare la forma IF ... THEN ... ELSE. Ecco un esempio:

```
10 INPUT ANNI
20 IF ANNI <18 THEN PRINT
   "MINORENNE" ELSE PRINT
   "MAGGIORENNE"
```

Il funzionamento è molto semplice: se il valore immesso è minore di 18, appare il messaggio "MINORENNE", altrimenti il messaggio "MAGGIORENNE".

La struttura IF ... THEN ... ELSE è molto utile nella stesura dei programmi, facilitandone la comprensione, pur non essendo strettamente necessaria. Se un particolare BASIC non è dotato di IF ... THEN ... ELSE, esistono due modi per aggirare l'ostacolo. Il primo impiega IF ... THEN, seguita da una GOTO per 'saltare' a un'altra parte del programma. L'ultimo programma presentato avrebbe questa forma:

```
10 INPUT ANNI
20 IF ANNI<18 THEN PRINT
   "MINORENNE": GOTO 30
25 PRINT "MAGGIORENNE"
30 ...resto del programma
```

Il secondo metodo impiega una seconda IF ... THEN, onde assicurarsi che ogni possibile condizione sia prevista:

```
10 INPUT ANNI
20 IF ANNI<18 THEN PRINT "MINORENNE"
25 IF ANNI>=18 THEN PRINT
   "MAGGIORENNE"
30 ...resto del programma
```

GRAFICA PER GIOCHI DA 10 MINUTI

Non occorre conoscere il codice macchina e neppure capire il sistema binario per fare un po' di grafica nei giochi. Ecco una serie di brevi programmi per tutti i computer

Chiunque, con una decina di ore di pratica su un computer, può creare caratteri grafici originali da usare nei giochi. Tutto ciò che serve è carta e matita per fissare le idee, più due o tre semplici routine per trasformare le idee in disegni del computer.

Ogni home computer ha il proprio metodo per la creazione di nuovi simboli grafici UDG (ossia definiti dall'utente). Anche le dimensioni dei caratteri che si possono creare variano da apparecchio ad apparecchio. Nel Commodore, per esempio, gli 'sprite' *standard* sono di 24 × 21 pixel (punti), mentre il massimo, per uno Spectrum, è di 8 × 8 pixel.

Ma, in ogni caso, conviene sempre partire con la creazione di simboli con dimensioni 8 × 8, quelle adottate, generalmente, per i 'nemici' nei giochi spaziali. Una volta che si è presa la mano in questo, è facile creare immagini grafiche di maggiori dimensioni, se il computer lo permette. In caso contrario, si possono unire più simboli per formarne uno solo.

DAL DISEGNO AL BINARIO

Il computer memorizza tutte le informazioni nel formato binario (ossia in base 2). Tuttavia, non è necessario conoscere il sistema binario per trasformare i caratteri creati su carta quadrettata in una serie di numeri binari. Quanto occorre sapere è:
1 Ogni volta che si vuole illuminare *un punto* si associa ad esso il numero 1.
2 Ogni volta che si vuole uno *spazio*, si usa uno Ø.

Si osservi, ad esempio, la croce di Lorena riportata più sotto. La riga superiore consiste di tre spazi, un punto e quattro spazi. In binario: ØØØ1ØØØØ. La seconda riga è composta da due spazi, tre punti e tre spazi, ossia ØØ111ØØØ: L'intera immagine si può rappresentare così:

Ø	Ø	Ø	1	Ø	Ø	Ø	Ø
Ø	Ø	1	1	1	Ø	Ø	Ø
Ø	Ø	Ø	1	Ø	Ø	Ø	Ø
Ø	1	1	1	1	1	Ø	Ø
Ø	Ø	Ø	1	Ø	Ø	Ø	Ø
Ø	Ø	Ø	1	Ø	Ø	Ø	Ø
Ø	Ø	Ø	1	Ø	Ø	Ø	Ø
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Su alcuni computer, i dati che servono alla costruzione di nuove immagini si possono immettere direttamente nel formato binario. Su altri, invece, occorre prima convertire i numeri in decimale (base 1Ø) o esadecimale (base 16).

Prima di operare qualsiasi conversione, si legga la sezione per il proprio computer (pagine 4Ø-44).

DAL BINARIO AL DECIMALE

La via più breve, per convertire i numeri binari in decimali (cioè le comuni unità, decine, centinaia e migliaia), è usare un piccolo grafico largo otto colonne e alto nove. Come intestazione, si scrivono i seguenti valori: 128, 64, 32, 16, 8, 4, 2, 1. Nelle restanti otto righe si riportano i valori binari associati all'immagine da creare. Ecco, per esempio, il grafico per la croce di Lorena:

128	64	32	16	8	4	2	1
Ø	Ø	Ø	1	Ø	Ø	Ø	Ø
Ø	Ø	1	1	1	Ø	Ø	Ø
Ø	Ø	Ø	1	Ø	Ø	Ø	Ø
Ø	1	1	1	1	1	Ø	Ø
Ø	Ø	Ø	1	Ø	Ø	Ø	Ø
Ø	Ø	Ø	1	Ø	Ø	Ø	Ø
Ø	Ø	Ø	1	Ø	Ø	Ø	Ø
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

Per fare la conversione, si ignorino tutti gli zeri. Per ciascuna riga orizzontale, si sommino i numeri riportati in cima alle colonne dove compaiono gli 1 binari. Nell'esempio proposto, la prima riga consiste in: nulla, nulla, nulla, 16, nulla, nulla, nulla, nulla. Totale della riga: 16.

La seconda riga, invece, consiste di: nulla, nulla, 32, 16, 8, nulla, nulla, nulla. Totale (32 + 16 + 8) = 56.

Completato il calcolo per tutte le righe, si ottengono 8 valori decimali e la frase DATA da immettere nel computer assomiglierà a questa:

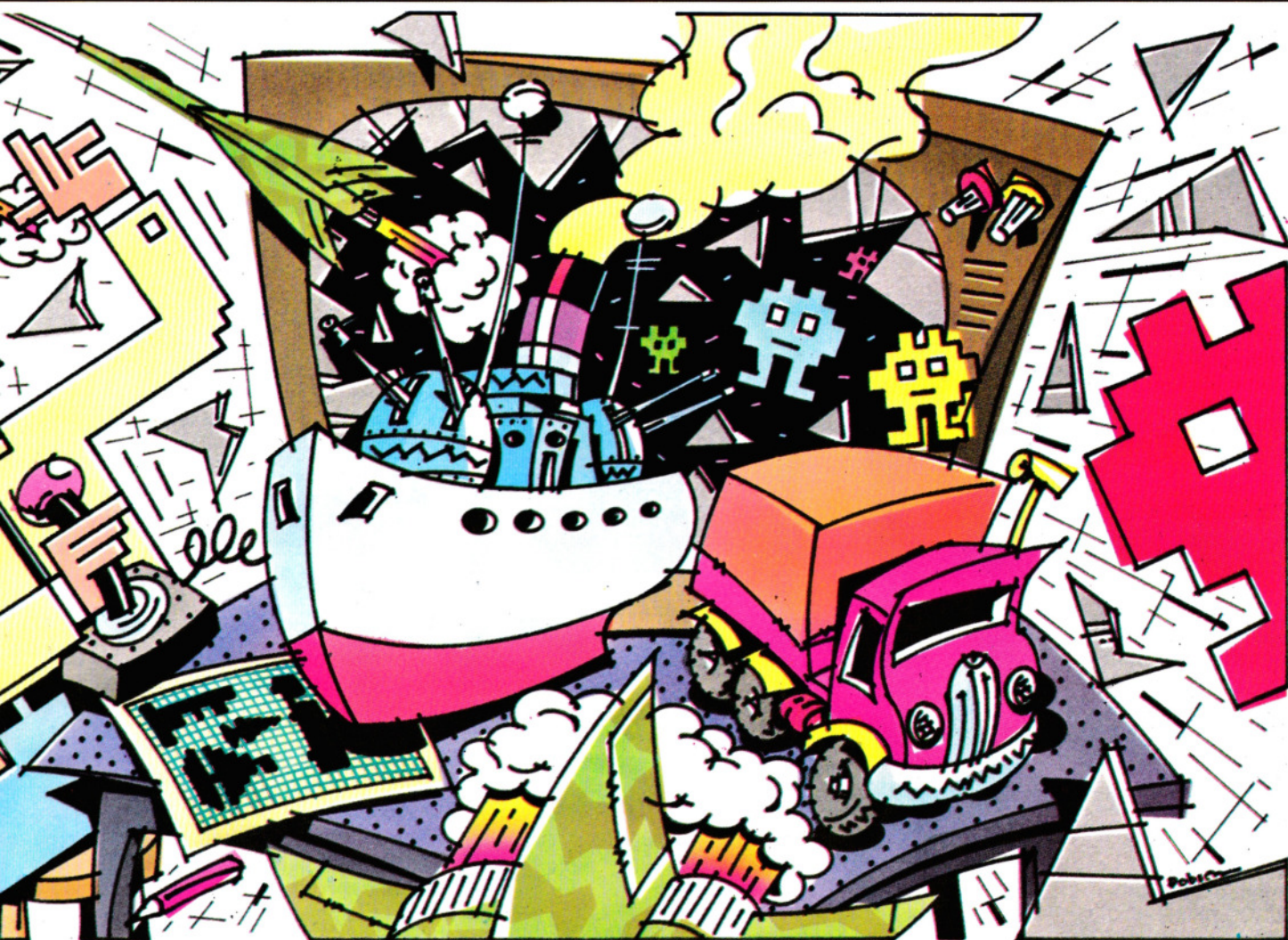
```
DATA 16, 56, 16, 124, 16, 16, 16, Ø
```

Inizialmente, può sembrare un metodo noioso, ma dopo qualche prova la conversione procede con rapidità. Inoltre, qualche particolare combinazione ricorrente (ad esempio 11111111, ossia 255 decimale) è facile da ricordare, senza che sia necessario doverla calcolare.

DAL BINARIO ALL'ESADECIMALE

La conversione da binario in esadecimale, se è ciò che richiede il computer, è addirittura più facile. La seguente tabella può essere utile. Non occorre, questa volta, annotare i numeri binari.

Binario	Esadecimale
ØØØØ	Ø
ØØØ1	1
ØØ1Ø	2
ØØ11	3
Ø1ØØ	4
Ø1Ø1	5
Ø11Ø	6

- DAL DISEGNO AL BINARIO IN MODO SEMPLICE
- CONVERSIONI BINARIO/DECIMALE E BINARIO/ESADECIMALE
- LA GRAFICA NEI PROGRAMMI

Ø111	7
1ØØØ	8
1ØØ1	9
1Ø1Ø	A
1Ø11	B
11ØØ	C
11Ø1	D
111Ø	E
1111	F

In questo caso *non* vanno ignorati gli zeri. La prima cosa da fare è dividere ogni linea di valori binari in due metà; si cerca nella tabella l'equivalente esadecimale delle prime quattro cifre binarie, lo si annota e si ripete l'operazione con la restante metà binaria. Il numero ottenuto è l'equivalente hex (abbreviazione per esadecimale).

Ritorniamo alla croce di Lorena. La prima mezza riga del binario ci è ormai familiare: ØØØ1. Dalla tabella, l'equivalente hex è 1; la seconda mezza riga è ØØØØ, ossia Ø hex. Scritti assieme, i due valori valgono 1Ø, il numero hex richiesto.

Analogamente, per la seconda riga: ØØ11 vale 3 hex, mentre 1ØØØ vale 8 hex. Il risultato è 38 hex.

Ripetendo il procedimento per tutti i valori binari, si ottiene un elenco di numeri hex da inserire nella frase DATA, che assomiglierà a:

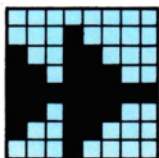
```
DATA 1Ø, 38, 1Ø, 7C, 1Ø, 1Ø, 1Ø, ØØ.
```

L'unica cosa che rimane da fare è comunicare al computer che i valori impostati sono esadecimali. Si vedano, a tal proposito, le sezioni relative ai singoli apparecchi.

Naturalmente, è possibile scrivere un programmino per la conversione da binario in decimale (o esadecimale), ma esso sarebbe di scarsa utilità se, contemporaneamente, si stanno scrivendo le frasi DATA di un altro programma.

Sia il Dragon che il Tandy accettano frasi DATA in hex, decimale o binario.

Se i dati sono in hex, va aggiunta semplicemente una linea al programma (si veda l'esempio).

Il primo programma disegna un piccolo aereo (vedi sotto) nell'angolo in alto a sinistra sullo schermo. Non è il miglior posto per osservarlo, ma certamente è il più facile da cui partire per farlo muovere sullo schermo.

```
20 PMODE 4,1
30 PCLS
40 SCREEN 1,1
60 FOR L=0 TO 7
70 READ N$
80 POKE L*32+1536,VAL("&H"+N$)
90 NEXT L
110 GOTO 110
500 DATA 00,10,18,9C,FF,9C,18,10
```

Si trascriva e si esegua un RUN

PMODE 4, 1 è stato scelto, nella linea 2Ø, poiché soltanto questo modo ad alta risoluzione permette la creazione di UDG.

Per ripulire lo schermo prima del disegno, si usa PCLS, alla linea 3Ø. Ciò vale per tutti i modi grafici ad alta risoluzione, non solo per il PMODE 4,1.

Per usare l'UDG sullo schermo in alta risoluzione, si impiega, nella linea 4Ø, l'istruzione SCREEN 1,1 che seleziona anche una colorazione in bianco e nero.

Il ciclo FOR ... NEXT, nelle linee 6Ø e 9Ø, fa sì che la linea 7Ø venga eseguita 8 volte. Ad ogni nuova esecuzione di READ N$, viene letta una nuova porzione della frase DATA (linea 5ØØ).

La linea 8Ø è importante per due ragioni. In primo luogo, fa apparire sullo schermo il modello di pixel che corrisponde al numero hex della linea 5ØØ. In secondo luogo, converte le informazioni della frase DATA in decimali, depositando i risultati direttamente nella parte di memoria del computer riservata alla visualizzazione sullo schermo.

Infine, la linea 11Ø serve unicamente come ciclo senza fine, per mantenere lo schermo in alta risoluzione. Omettendo questa linea, il Dragon tornerebbe automaticamente allo schermo testuale, cancellando il disegno appena creato.

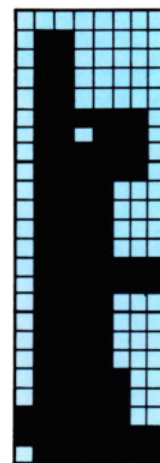
UDG PIU' ALTI

Semplicemente cambiando la linea 6Ø e variando i valori nella DATA (linea 5ØØ), si può creare un UDG alto e stretto, invece che di 8×8 pixel.

Si modifichi la linea 6Ø in:

```
60 FOR L=0 TO 23
```

Poi si cambi la linea 5ØØ in:

```
500 DATA 00,60,60,60,60,7E,6E,7E,7E,78,78,
    78,78,7F,7F,78,78,78,78,7C,7C,FC,FF,7F
```

Avviato il programma, sullo schermo si vedrà l'immagine di un coniglio.

L'aver cambiato la linea 6Ø consente di leggere un maggior numero di dati dalla linea 5ØØ, che adesso contiene i valori esadecimali necessari per creare la forma del coniglio.

Con questo sistema, si possono creare UDG di qualsiasi altezza. Una volta disegnato su carta l'UDG, si contano quante linee di pixel occorrono. Si modifica di conseguenza la linea 6Ø, per la lettura del numero corretto di dati e si trascrivono i dati nella linea 5ØØ assicurandosi che il numero di dati corrisponda al numero di volte che viene eseguito il ciclo FOR ... NEXT.

UDG PIU' LARGHI

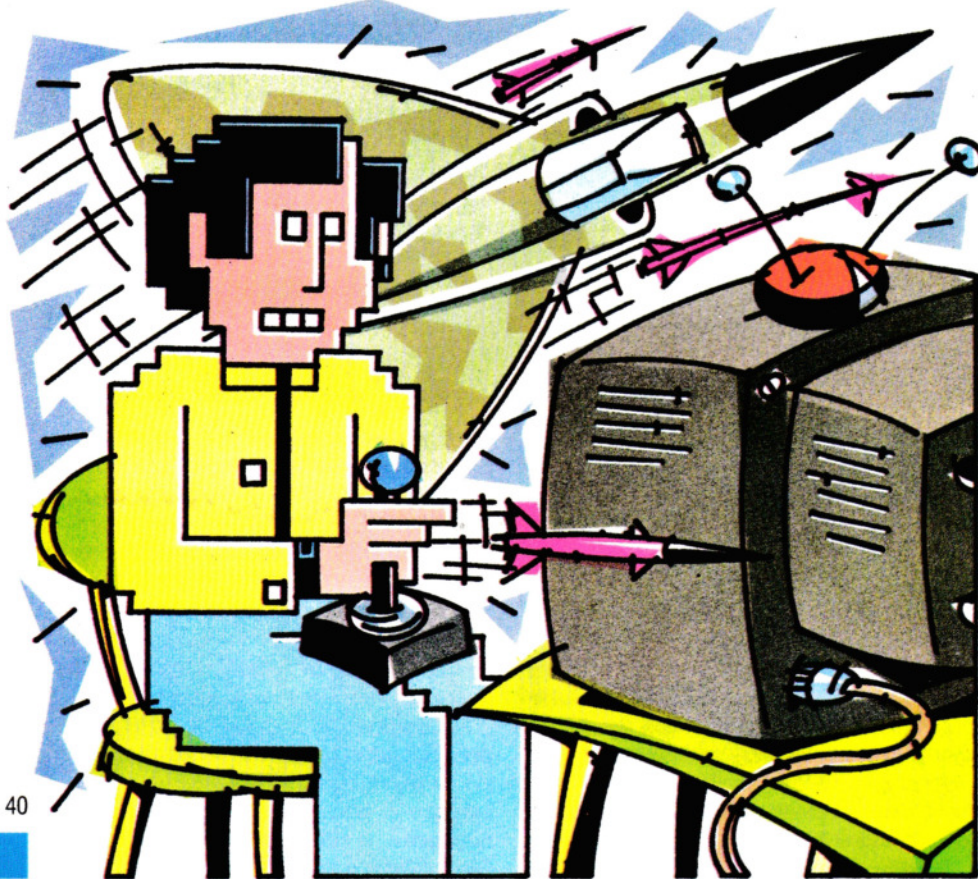
Produrre un'immagine più larga e bassa, invece che più alta e stretta, è leggermente più complesso.

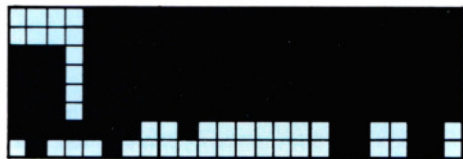
Si parte cambiando le linee 6Ø ed 8Ø:

```
60 FOR L=0 TO 7
80 POKE L*32+1536+F,VAL("&H"+N$)
```

Poi si riscriva la linea 5ØØ:

```
500 DATA 0F,0F,EF,EF,EF,EF,FE,44,FF,FF,FF,
    FF,FF,FF,40,00,FF,FF,FF,FF,FF,FF,66,66
```

Infine, si aggiungano le linee:

```
50 FOR F=0 TO 2
100 NEXT F
```

Quando si avvia il programma, sullo schermo appare l'immagine di un camion articolato.

Come funziona? I 24 dati, alla linea 500, formano tre quadrati, ciascuno alto 8 pixel.

Se la frase DATA venisse letta con un solo ciclo FOR ... NEXT, come nel programma precedente, l'immagine sarebbe quantomeno bizzarra ... tagliata in tre fette sovrapposte l'una all'altra.

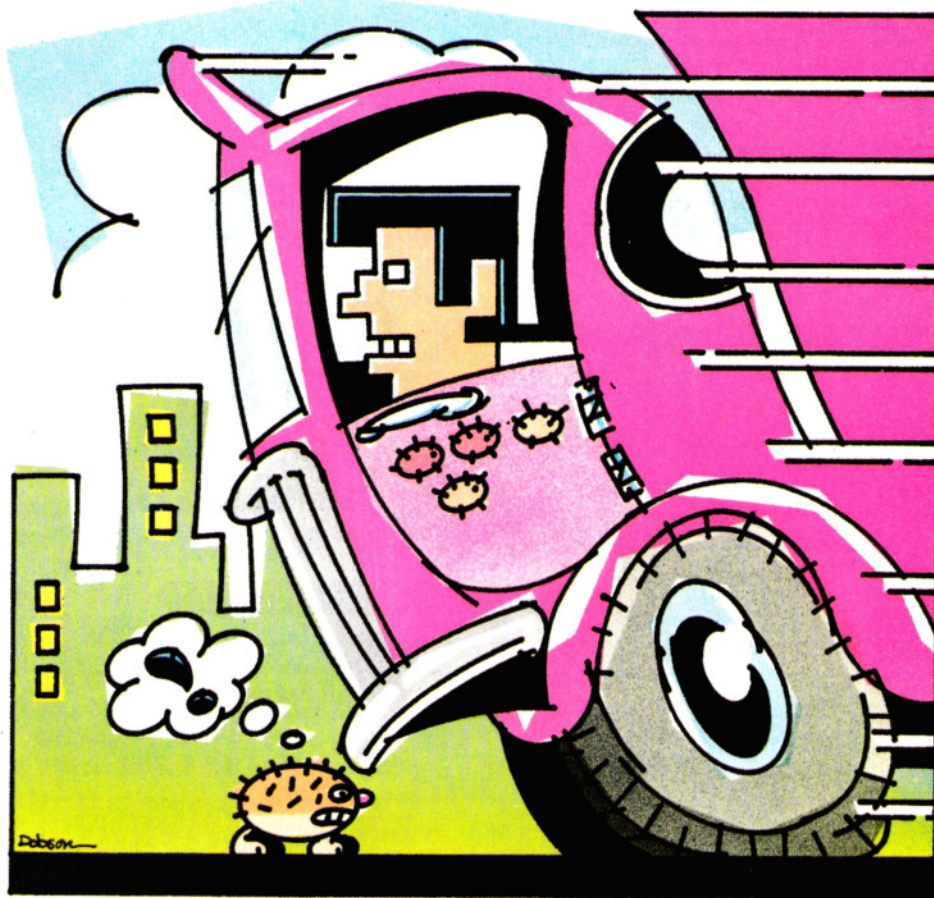
Per avvertire il computer che i tre quadrati devono essere affiancati, viene usato un secondo ciclo FOR ... NEXT (linee 50 e 100) e la linea 80 è modificata in modo da tener conto del valore di F nell'istruzione POKE.

Così il programma legge i primi otto dati e li visualizza sullo schermo. Il nuovo ciclo FOR ... NEXT, mutando il valore della POKE, fa sì che il secondo blocco di dati venga visualizzato accanto al primo. Infine, viene letto e visualizzato il terzo blocco di dati, di fianco ai primi due.

UDG IN MOVIMENTO

Per muovere sullo schermo l'immagine dell'aereo, si aggiungano queste linee:

```
110 DIM A(3),B(3)
120 GET (0,0)-(7,7),A,G
130 PCLS
140 LET X=127
150 LET Y=95
160 PUT (X,Y)-(X+7,Y+7),A,PSET
170 LET LX=X
180 LET LY=Y
190 IF PEEK(338)=239 AND Y>2 THEN Y
    =Y-2:GOTO 240
200 IF PEEK(342)=247 AND Y<182 THEN
    Y=Y+2:GOTO 240
210 IF PEEK(340)=223 AND X>3 THEN X
    =X-3:GOTO 240
220 IF PEEK(338)=223 AND X<245 THEN
    X=X+3:GOTO 240
230 GOTO 190
240 PUT (LX,LY)-(LX+7,LY+7),B,PSET
250 GOTO 160
```

Sul Tandy, cambiare il valore 239 (linea 190) in 251: il valore 247 (linea 200) in 253; il valore 223 (linee 210 e 220) in 247.

Dopo un RUN, i tasti Z, X, P e L provocano rispettivamente, un movimento verso sinistra, verso destra, verso l'alto e verso il basso.

In questo programma vengono presentate tre nuove parole chiave: GET, PUT e DIM. Esse verranno spiegate ampiamente in seguito. Brevemente, GET serve per memorizzare una parte dello schermo, PUT per depositarvi dei valori e DIM per riservare una parte di memoria all'operazione GET.

Le linee da 190 a 220 servono per individuare quale tasto sia stato premuto e per muovere l'UDG. A questo proposito, si veda a pagina 59. Per muovere il coniglio apportare le seguenti modifiche (sul Tandy si usi 253 e non 247 alla linea 200):

```
110 DIM A(6),B(6)
120 GET (0,0)-(7,23),A,G
160 PUT (X,Y)-(X+7,Y+23),A,PSET
200 IF PEEK(342)=247 AND Y<166 THEN
    Y=Y+2:GOTO 240
240 PUT (LX,LY)-(LX+7,LY+23),B,PSET
```

Per muovere il camion, apportare le seguenti modifiche (sul Tandy cambiare 247 in 253, alla linea 200, e 223 in 247, alla linea 220):

```
110 DIM A(6),B(6)
120 GET (0,0)-(23,7),A,G
160 PUT (X,Y)-(X+23,Y+7),A,PSET
200 IF PEEK(342)=247 AND Y<182 THEN
    Y=Y+2:GOTO 240
220 IF PEEK(338)=223 AND X<229 THEN
    X=X+3:GOTO 240
240 PUT (LX,LY)-(LX+23,LY+7),B,PSET
```

USO DI DATI BINARI

Si possono anche usare valori binari nelle frasi DATA, purché ogni numero sia composto da 8 cifre (un byte). Si devono anche aggiungere le seguenti linee:

```
71 LET N=0
72 FOR J=1 TO 8
74 IF MID$(N$,J,1)="1" THEN N=N+2↑
   (8-J)
76 NEXT J
```

assicurandosi che la linea 80 sia:

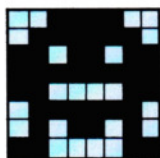
```
80 POKE L*32+1536+F,N
```

Queste nuove linee esaminano a turno ciascun bit del numero e lo convertono in numero decimale. Si tratta, in breve, dell'equivalente computerizzato della tabella vista all'inizio della lezione.

- Il movimento di immagini in alta risoluzione verrà trattato più diffusamente in seguito.

Sulle macchine Acorn l'elenco dei numeri che definiscono i caratteri speciali può essere, espresso sia in decimale che in esadecimale. Di solito, è più facile, consultando la tabella a pagina 38, convertire direttamente i valori binari in hex, piuttosto che trasporre il tutto in numeri decimali. Si ricordi, comunque, che ogni numero hex deve iniziare col carattere &, in modo da comunicare al computer che tipo di numero stiamo usando.

I valori per ottenere il piccolo marziano (vedi sotto) sono:

Binario	Hex	Decimale
00111100	3C	60
01111110	7E	126
11011011	DB	219
11111111	FF	255
11000011	C3	195
01111110	7E	126
01011010	5A	90
11000011	C3	195

Cosicché, per definire il carattere si usa:

```
10 VDU 23,224,&3C,&7E,&DB,&FF,&C3,&7E,
   &5A,&C3
```

oppure:

```
10 VDU 23,224,60,126,219,255,195,126,90,
   195
```

La parte VDU 23 significa 'definisci un carattere' e il numero che segue è il codice associato al nuovo carattere. Sono disponibili 32 numeri di codice, numerati da 224 a 255. Non ha importanza l'ordine in cui si usano. Dopo il codice, vengono gli otto valori che erano stati calcolati per il piccolo marziano.

Abbiamo definito il carattere e il computer sa cos'è, ma come visualizzarlo sullo schermo? Questo è facile: si seleziona uno dei modi grafici (non il 7).

Per esempio:

```
5 MODE 1
```

Adesso, basta usare l'istruzione PRINT:

```
20 PRINT TAB (5,10) CHR$ 224
```

Eseguendo un RUN, si vedrà il risultato. La funzione CHR$ serve per visualizzare un carattere fornendone il codice (nel nostro caso 224). Poiché CHR$, di per sé, non dice molto, possiamo aggiungere la linea:

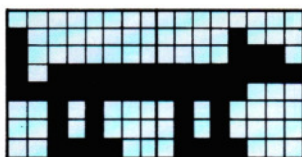
```
15 marziano $=CHR$ 224
```

e modificare la linea 20 in:

```
20 PRINT TAB (5,10) marziano $
```

Adesso, il programma è più chiaro.

CARATTERI PIU' GRANDI

L'immagine che segue, un bassotto, è larga ma poco alta e occupa due griglie. Occorre quindi definire due VDU:

```
100 VDU 23,225,&00,&80,&80,&BF,&FF,&28,
    &28,&3C
110 VDU 23,226,&00,&08,&0E,&FF,&F8,&50,
    &50,&7C
```

Poi le due parti devono essere visualizzate l'una a fianco all'altra:

```
120 cane$=CHR$ 225+CHR$ 226
130 PRINT TAB(10,10) cane$
```

Si noti che, usando PRINT TAB, si può visualizzare il carattere dove si desidera.

Per un bassotto più lungo, occorre un'altra definizione di carattere:

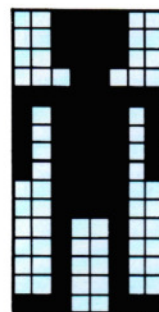
```
115 VDU 23,227,&00,&00,&00,&FF,&FF,&00,
    &00,&00
```

Adesso si aggiunge il nuovo carattere:

```
120 cane$=CHR$225+CHR$227+
    CHR$226
```

Ora si dia un RUN per verificare la forma. Volendo, si possono aggiungere ancora altre sezioni al cane, fino a riempire tutta la larghezza dello schermo!

UN CARATTERE ALTO E STRETTO

Per visualizzare l'omino qui riprodotto, occorre prima definirlo:

```
200 VDU 23,228,&3C,&3C,&3C,&18,&FF,&BD,
    &BD,&BD
210 VDU 23,229,&BD,&3C,&3C,&24,&24,&24,
    &24,&E7
```

L'accorgimento, qui, è di visualizzare le due metà esattamente l'una sopra all'al-

tra e ciò si ottiene in due modi. Il primo consiste nel definire con precisione la posizione nel comando PRINT:

```
220 PRINT TAB(15,9) CHR$228: PRINT
    TAB(15,10) CHR$229
```

Si usi un RUN per verificare il risultato. L'omino dovrebbe essere collocato accanto al cane.

Ma un metodo migliore è di definire l'immagine come si è fatto per il cane:

```
220 uomo$=CHR$228+CHR$10+CHR$8
    +CHR$229
230 PRINT TAB(15,9) uomo$
```

Le istruzioni CHR$10 e 8 controllano la posizione del cursore: la prima lo sposta in basso di una riga, la seconda indietro di una posizione (esattamente sotto la prima metà). Le tre immagini sono state create una alla volta, ma normalmente tutte le varie definizioni con le VDU 23 sono raggruppate all'inizio del programma, in modo da renderlo più ordinato e chiaro.

Ecco di seguito un elenco di tutte le linee usate fin qui, ma opportunamente rinumerate per chiarezza.

```
5 MODE 1
10 VDU 23,224,&3C,&7E,&DB,&FF,&C3,&7E,
   &5A,&C3
20 VDU 23,225,&00,&80,&80,&BF,&FF,&28,
   &28,&3C
30 VDU 23,226,&00,&08,&0E,&FF,&F8,&50,
   &50,&7C
40 VDU 23,227,&00,&00,&00,&FF,&FF,&00,
   &00,&00
50 VDU 23,228,&3C,&3C,&3C,&18,&FF,&BD,
   &BD,&BD
60 VDU 23,229,&BD,&3C,&3C,&24,&24,&24,
   &24,&E7
70 marziano$=CHR$224
80 cane$=CHR$225+CHR$227+CHR$226
90 uomo$=CHR$228+CHR$10+CHR$8+
   CHR$229
100 PRINT TAB(5,10) marziano$
110 PRINT TAB(10,10) cane$
120 PRINT TAB(15,9) uomo$
```

I caratteri UDG, sul Commodore 64, si usano meno frequentemente dei più versatili sprite (si veda a pagina 15).

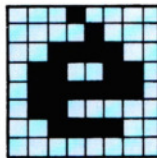
Ad ogni modo, saper definire dei caratteri UDG a 8×8 pixel può tornare utile, specialmente per ridefinire in parte o tutto, l'insieme dei caratteri normali (può essere utile, ad esempio, creare lettere speciali allo scopo di comporre messaggi in più lingue).

Un carattere quale la 'è' (e accentata), che è comunissima in italiano, non esiste nel repertorio dei caratteri di normale impiego del Commodore.

Questo e altri nuovi caratteri possono esser definiti, nei programmi, per una migliore visualizzazione di testi.

Per la creazione di un UDG si seguono gli stessi metodi visti fin qui per le altre macchine.

Per esempio, il carattere 'è' viene definito dai seguenti valori:

Binario	Hex	Decimale
00010000	10	16
00001000	08	8
00111100	3C	60
01100110	66	102
01111110	7E	126
01100000	60	96
00111100	3C	60
00000000	00	0

Lo schema decimale è quello di uso più immediato, ma scomodo da calcolare. Con qualche accorgimento, però, si possono usare anche valori binari o esadecimali.

Proviamo a includere questo carattere in un nuovo set di caratteri. Si trascriva:

```
10 A=12:Z=A*1024/256
20 POKE 53272, (PEEK(53272)
   AND240) OR A
30 POKE 52,Z: POKE 56,Z: CLR: A=12
40 POKE 56334, PEEK (56334) AND 254
50 POKE 1, PEEK (1) AND 251
60 FOR J=0 TO 56832-53248
70 POKE A*1024+J, PEEK (53248+J)
80 NEXT J
90 POKE 1, PEEK (1) OR 4
100 POKE 56334, PEEK (56334) OR 1
110 SC=5: Z=1024*12: FOR J=Z+
   (SC*8) TO Z+(SC*8)+7: READ A$
120 N=0: FOR T=1 TO LEN(A$)
130 IF MID$(A$,T,1)="1" THEN N=N+2↑
   (LEN(A$)-T)
140 NEXT T:POKE J,N: NEXT J
500 DATA 00010000
510 DATA 00001000
520 DATA 00111100
530 DATA 01100110
540 DATA 01111110
550 DATA 01100000
560 DATA 00111100
570 DATA 00000000
```

Lanciando il programma, non accade niente per circa un minuto, ma quando appare il segnale di 'pronto', si prema il tasto [E]. Sullo schermo compare una è. Se si dovessero vedere soltanto simboli grafici, si premano contemporaneamente i tasti [C=] e [SHIFT].

Adesso si provi a variare la serie di 1 e Ø della frase DATA, per creare nuovi caratteri o simboli grafici. Per verificarne l'effetto, si rilanci il programma e si prema nuovamente il tasto [E] per visualizzare il nuovo carattere.

QUALCHE DETTAGLIO

Per chi ha un po' più d'esperienza, ecco una spiegazione sul funzionamento del programma.

I caratteri normali, molti dei quali si possono anche riutilizzare per nuovi insiemi, sono nelle ROM e non si possono quindi cambiare. La tecnica consiste nel ricopiare nelle RAM, in tutto o in parte, il set di caratteri contenuto nelle ROM e apportare le modifiche volute. I nuovi caratteri possono essere lettere o simboli o un misto fra i due. A questo punto, si istruisce il computer a usare i nuovi caratteri e non quelli contenuti nelle ROM.

Il programma svolge una serie di operazioni ben distinte, la prima delle quali è quella di assegnare memoria RAM sufficiente per contenere il nuovo set di caratteri. Il valore di A (linee 1Ø e 3Ø), permette di scegliere l'area di memoria da usare. Si può impiegare un qualsiasi valore intero compreso tra 4 e 16. Nel programma viene usato 12, il che sposta il puntatore della memoria caratteri a 12288 (cioè 12 * 1Ø24). Un valore di 4 collocherebbe il deposito alla locazione 4Ø96 (ossia 4 * 1Ø24), e via dicendo.

La linea 3Ø sposta due puntatori (quello di fine-BASIC e quello all'inizio dell'area stringhe), in modo che i programi BASIC non vadano a sovrapporsi al nuovo set di caratteri, danneggiandolo.

Poi, alla linea 4Ø, il programma disabilita la cosiddetta 'scansione ad interrupt' della tastiera. La linea 5 seleziona la ROM dei caratteri.

La routine di copiatura interviene alle linee 6Ø, 7Ø e 8Ø. I numeri 53248 e 56832 (linea 6Ø) si riferiscono agli indirizzi iniziali e finali dei gruppi di otto blocchi di caratteri, da copiare in RAM. Benché il programma esegua una copia integrale, è anche possibile restringere questa operazione al minimo indispensabile, cambiando l'arco di valori attribuito a J. Si può addirittura selezionare e modificare un solo carattere per volta, come vedremo oltre.

Quando la copiatura è completa, il generatore di caratteri ROM viene disabilitato (linea 9Ø), mentre viene riabilitata la scansione della tastiera (linea 1ØØ).

Le linee 11Ø e 14Ø cambiano la definizione del un carattere scelto (SC) in memoria, leggendo la serie di valori contenuti nelle frasi DATA (linee 5ØØ-58Ø) e convertendoli in valori decimali (linee 12Ø e 13Ø. Il valore di SC è quello necessario alla POKE su video, come si può verificare sui manuali d'uso. Un SC pari a 5, ad esempio, visualizza il nuovo carattere quando viene premuto il tasto [E]. Si provi a cambiare questo valore e a rilanciare il programma utilizzando un'altra definizione di carattere UDG.

Per risparmiare spazio, le linee DATA si possono comprimere in una sola linea (omettendo, ovviamente le linee 51Ø-58Ø):

```
500 DATA 00010000,00001000,00111100,
    01100110,01111110,01100000,
    00111100,00000000
```

Si noti, però, che questo formato non permette di valutare facilmente l'aspetto dell'UDG, né di correggerlo rapidamente.

Se si desidera lavorare in notazione hex, si apportino le seguenti modifiche, cancellando le linee da 51Ø a 58Ø:

```
130 M=ASC(MID$(A$,T,1))-48:N=(M+(M
    >9)*7)*16↑(LEN(A$)-T)+N
500 DATA 10,08,3C,66,7E,60,3C,00
```

Ciò rende molto più semplice digitare o correggere la linea 5ØØ.

S

Lo Spectrum accetta frasi DATA in formato sia binario che decimale, ma non esadecimale. Per vedere come procedere, si scriva:

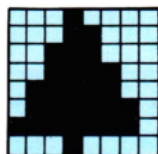
```
PRINT "(A grafica)"
```

Per impostare (A grafica), si premano contemporaneamente i tasti [CAPS SHIFT] e [GRAPHICS], poi si digiti una A.

Apparentemente, si vedrà solo una comune A maiuscola. Ma, come si è già spiegato a pagina 8, essa può essere ridefinita in una qualsiasi forma 8×8.

E per farlo, basta un programma di sole cinque linee. Per ottenere l'alberino qui a fianco, per esempio, si scriva:

```
10 FOR n=0 TO 7
20 READ dato
30 DATA BIN 00010000, BIN 00011000, BIN
   00111000, BIN 00111100, BIN
   01111100, BIN 01111110, BIN
   11111110, BIN 00010000
40 POKE USR "a"+N, dato
50 NEXT n
```

Questo programma usa un ciclo FOR ... NEXT (linee 1Ø e 5Ø) per richiamare le otto

righe di dati (linee 3Ø e 4Ø) da depositare in memoria. La linea 2Ø legge un dato alla volta, mentre la 4Ø lo deposita.

Si dia un RUN, poi si scriva, di nuovo:

```
PRINT "(A grafica)"
```

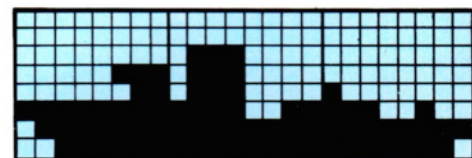
Si vedrà che la A maiuscola è svanita e al suo posto c'è ora un piccolo abete.

Se adesso si digita NEW, il programma verrà cancellato, ma l'immagine dell'abete resta definita in memoria fino allo spegnimento dell'apparecchio. Possiamo quindi muoverla a nostro piacimento o usarla per decorazione, come un normale carattere. Si provi questo:

```
5 CLS
10 FOR y=3 TO 19
20 LET x=INT (RND*20)+5
30 PRINT AT y,x; INK 4;"(A grafica)"
40 LET xx=INT (RND*20)+5
50 PRINT AT y,xx; INK 2; "(A grafica)"
60 NEXT y
```

Sono degli ostacoli per una gara di sci?

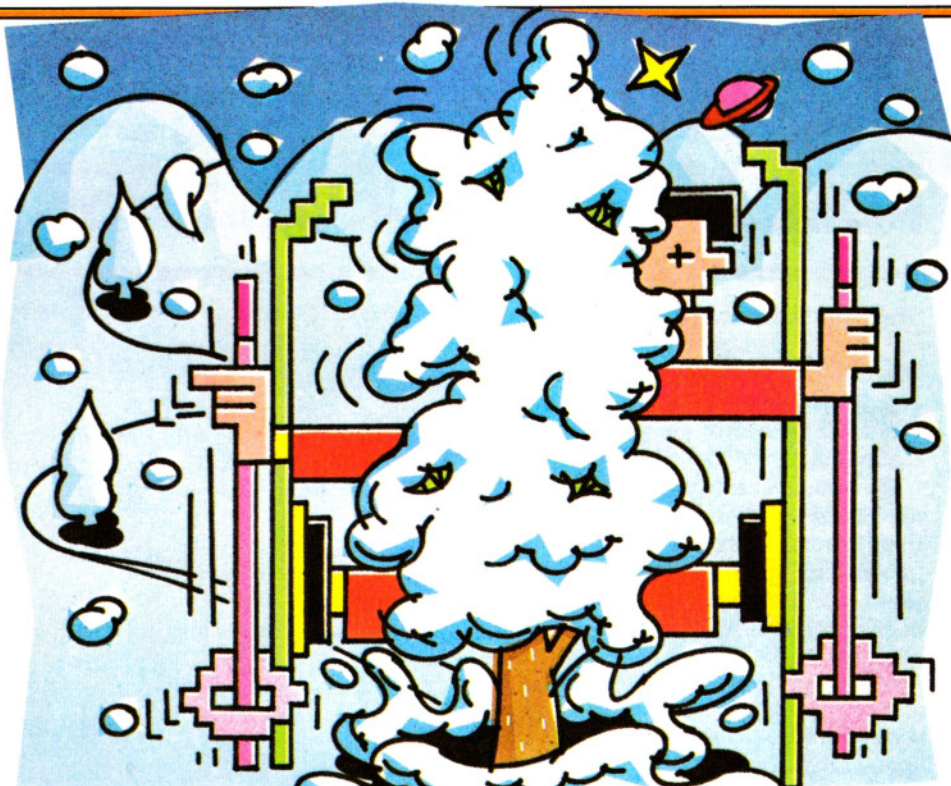
Volendo creare un carattere UDG di dimensioni maggiori di 8×8, basta richiamare e definire due o più UDG alla volta. Questo programma, ad esempio, crea la sezione di prua del cacciatorpediniere qui riprodotto che si potrebbe usare in un gioco (si veda in Giochi col Computer 1):

```
10 FOR n=0 TO 7
20 READ a
30 DATA BIN 0, BIN 0, BIN 0, BIN
   00000111, BIN 00000011, BIN
   11111111, 00000011, BIN 11111111,
   BIN 01111111, BIN 00111111
40 POKE USR "a"+n, a
50 NEXT n
```

Due cose vanno segnalate. La prima è che si può usare BIN Ø, al posto di otto Ø, se l'intera file è 'vuota'. La seconda è che nella linea 2Ø è stata usata la variabile

semplice a, invece della parola DATO, usata nel precedente programma per maggiore chiarezza. Si potrebbe egualmente usare b, c o x, o anche PIPPO, purché poi corrisponda a quella adoperata nella linea 40. Quando si arriva a inserire le DATA delle sezioni centrali e di poppa, non occorre ribattere l'intero programma. Con un solo RUN dato al programma e il programma ancora in memoria, basta correggere la linea 40, cambiando USR "a" in USR "b". Ovviamente, si deve inserire una nuova linea 30, con i nuovi dati.

Si faccia attenzione, nell'inserire i dati, che le righe siano sempre 8, anche se alcune sono solo zeri: meno righe provocano un errore del tipo E Out of DATA, 20:1, mentre un numero maggiore di righe... fa affondare la nave!

Infine, per inserire i dati in forma decimale e non binaria, prima occorre convertirli, come già descritto. Poi, nell'inserirli, si tralascia la parola BIN. La linea di DATA per il cacciatorpediniere diventerebbe:

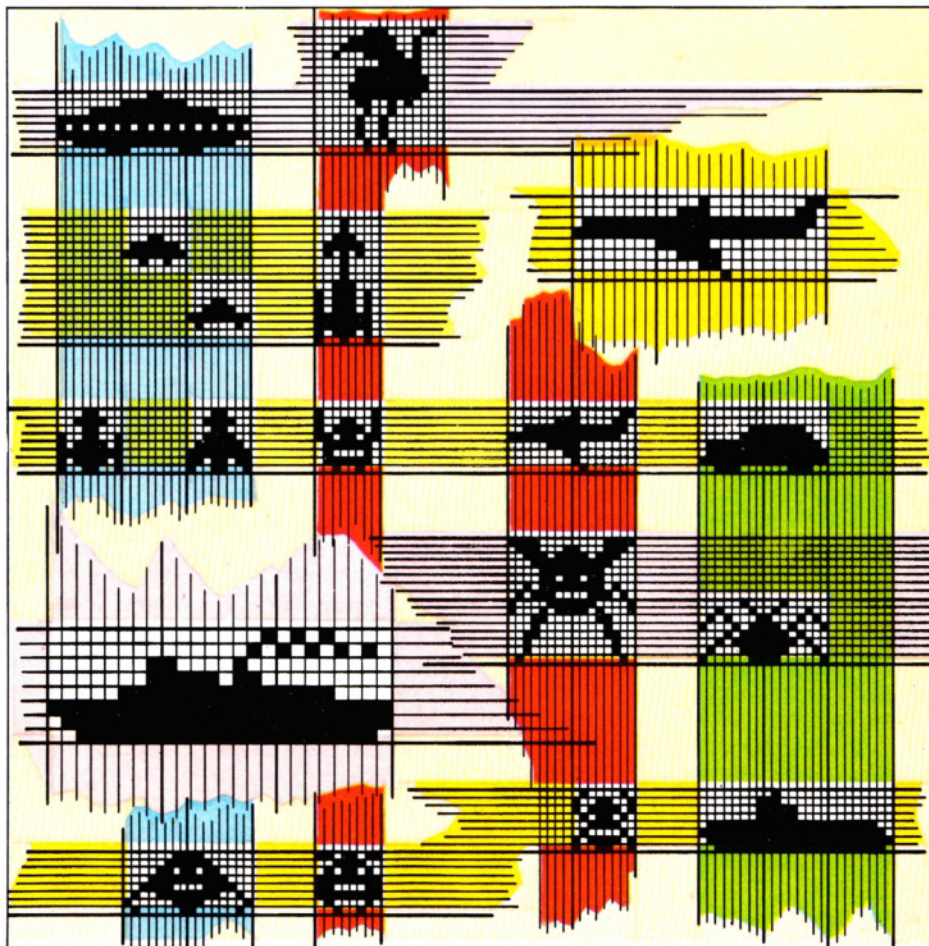
```
30 DATA 0, 0, 0, 7, 3, 255, 127, 63
```

E ORA

Una volta afferrati i principi generali, tutta la grafica di queste pagine può venir disegnata con successo su tutti gli apparecchi. Già dopo averne sperimentati alcuni, comunque, ci si sentirà capaci di disegnare immagini del tutto nuove.

Nell'immissione di frasi DATA, è meglio usare la notazione binaria, decimale oppure esadecimale?

Se è possibile scegliere tra binario e decimale, binario è meglio, perché consente di modificare anche un solo punto dell'immagine.

Il formato decimale può essere più facile da ricordare, se ricorre spesso nella definizione di un'immagine grafica.

Potendo scegliere tra decimale ed esadecimale, quest'ultimo è più immediato e offre un altro vantaggio: ci abitua al codice usato nel linguaggio macchina.

METTIAMO ORDINE NELL'ARCHIVIO DEGLI HOBBY

La vostra rubrica degli indirizzi è in completo disordine? C'è disorganizzazione nella gestione dell'archivio? Ecco alcuni metodi per risolvere simili problemi

'Ma in pratica, a che serve?' è la domanda che più frequentemente viene rivolta a chi possiede un home computer. 'A un po' di tutto...' è la risposta più ovvia, se vogliamo la più breve, ma non certo la più esauriente.

Ma ecco, in queste pagine, un programma davvero utile: un sistema di archiviazione talmente flessibile, da avere un'illimitata quantità di applicazioni. Lo si può usare per archiviare i nomi e gli indirizzi degli amici, o dei membri di un'associazione, oppure per catalogare una collezione di monete, di ricette o (perché no?) anche di *computer games*.

L'unico limite a ciò che si può ottenere con questo programma è imposto soltanto dalle dimensioni della memoria RAM disponibile.

Per la maggior parte dei casi, 32 Kbyte sono il minimo indispensabile, escludendo, quindi, l'impiego dello ZX81, dello Spectrum da 16K e del Vic20.

Comunque sia, va ricordato che, a causa delle contenute dimensioni della RAM negli home computer, conviene non eccedere nella lunghezza di ciascuna voce da registrare.

IL MENU PRINCIPALE

Una volta trascritto e *lanciato* il programma con un RUN, appare sul video il menu principale.

A questo punto, possiamo, per esempio, "immettere un record", oppure "operare una ricerca".

Ma, per prima cosa, dobbiamo "creare il file", ossia l'archivio dei dati.

LA CREAZIONE DEL FILE

Ai computer, però, è necessario impartire precise istruzioni su cosa desideriamo ottenere, prima di poter conseguire qualsiasi risultato concreto.

Per creare un file è necessario comunicare al computer il numero di record (schede) voluti, e la lunghezza massima di ogni record.

"CREARE UN FILE" è l'opzione n. 1 del menu, selezionabile mediante la pressione del tasto [1].

Compare la scritta "Sei sicuro?", per evitare che, a seguito di una pressione accidentale del tasto [1], si causi la distruzione di un archivio già esistente. Se si intende proseguire nella creazione, si prema il tasto [S], ma, qualora il file già contenga delle informazioni, basterà premere un qualsiasi altro tasto ([N] ad esempio) e il programma farà ritorno al menù principale.

LA LUNGHEZZA DEI CAMPI

Avendo optato per la creazione, il computer ci chiede ora il numero di campi desiderato. I *campi* sono gli elementi d'informazione che vanno immagazzinati in ciascun record.

A un collezionista di francobolli, per esempio, possono bastare quattro campi: data d'emissione, valore nominale, ultima valutazione e note.

Per risparmiare memoria, il numero massimo di campi, per ciascun record, è limitato a otto.

Immesso il valore desiderato, il programma chiede il 'Nome del primo campo' (nell'esempio prima citato, scriveremo "EMISSIONE").

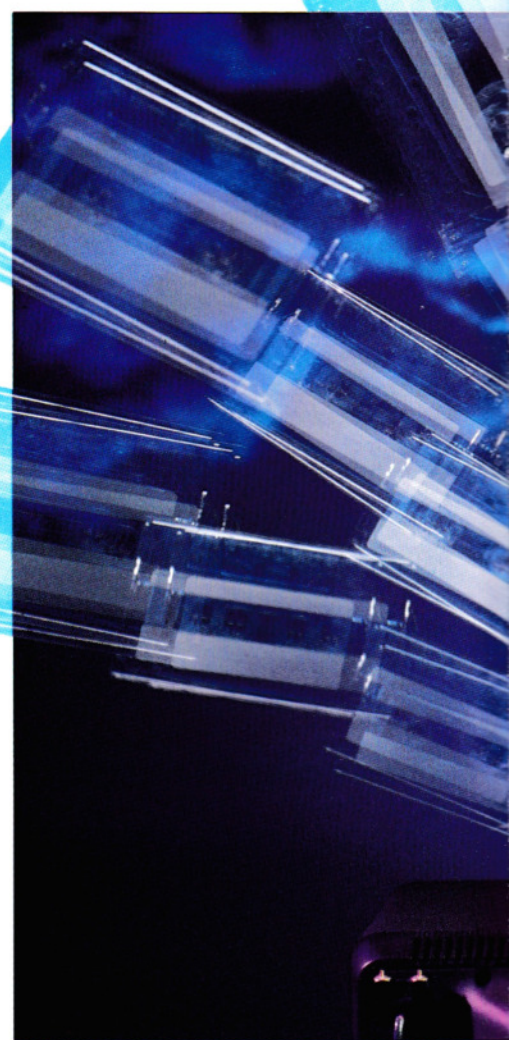
Viene poi richiesta la lunghezza del primo campo, ossia il numero massimo di caratteri che esso può contenere. La lunghezza massima è di 19 caratteri (27 nella versione Acorn).

Se prevediamo che, ad esempio, per un indirizzo sia necessario un numero maggiore di caratteri, dovremo suddividerlo in più campi (via, numero, C.A.P., e via dicendo).

Ottenute le risposte sul primo campo, il computer passa al secondo, poi al terzo, fino al numero da noi precedentemente indicato.

Ovviamente, più brevi sono i dati immessi, maggiore è la quantità di record che potremo archiviare.

Terminata questa fase, il computer calcola quanti record potremo memorizzare, visualizzando infine il risultato sullo schermo.

Solo nella versione Spectrum, questo valore va immesso manualmente, per evitare che nel corso della registrazione, l'apparecchio perda tempo a ricopiare parti di memoria che in realtà non vengono utilizzate.

IMMISSIONE DI UN RECORD

Completata la creazione del file, il programma torna al menu principale dal quale, attraverso la pressione del tasto [2], si sceglie l'opzione n. 2 (IMMISSIONE DI UN RECORD).

In alto, sullo schermo, appare un contatore dei record immessi, assieme a un messaggio dal titolo: "Hai usato 10 dei 100 record disponibili".

Sotto questa scritta, appaiono i nomi dei campi.

Possiamo a questo punto immettere le informazioni che desideriamo archiviare,

- TENERE IN ORDINE RACCOLTE DI OGGETTI
- UN PRATICO PROGRAMMA DI ARCHIVIAZIONE
- USO EFFICIENTE DELLA MEMORIA DISPONIBILE
- LA CREAZIONE DI UN FILE
- AGGIUNTE E MODIFICHE AL FILE
- CONSERVARE E RITROVARE LE INFORMAZIONI

sotto ciascuna voce.

È consigliabile cercare di limitare la lunghezza dei dati, per i motivi che abbiamo già esposti.

Premendo il tasto ENTER o RETURN, il dato che abbiamo immesso viene duplicato accanto al nome del campo e il cursore si riposiziona in attesa dell'immissione successiva.

Terminata l'immissione di tutti gli elementi di un record, si passa automaticamente a quello successivo, ma se premiamo RETURN (oppure ENTER) prima di digitare il dato relativo al primo campo, il programma ci porterà nuovamente al menu principale.

LA VISUALIZZAZIONE

Per rivedere i record immessi, si sceglie l'opzione n. 3 del menu "VISIONE RECORD", con la pressione del tasto 3.

Il programma visualizza il primo record (che non necessariamente è il primo da noi immesso, ma, semplicemente, il primo secondo un criterio di scelta diverso: quello alfabetico).

I metodi che sono usati dai computer per il riordino alfabetico variano di poco tra di loro.

In generale, l'ordinamento avviene in base al primo campo, il quale spesso contiene il 'NOME'.

Se il primo carattere di due campi è identico, il confronto passa al secondo carattere, e così via.

Il primo problema sorge quando nel primo campo si hanno numeri.

Infatti, il computer ordina i numeri delle lettere.

Ma, nel caso di soli numeri, il confronto procede cifra per cifra e non per il valore globale del numero. In altre parole, se nel primo campo abbiamo usato una numerazione da 1 a 100, l'ordinamento produrrà: 1, 10, 11, 12... fino a 100, prima di passare a 2, 20, 21, ecc.

La più ovvia soluzione è di usare degli zeri davanti ai numeri, ossia: 001, 002 e così di seguito.

Il secondo problema sorge se nel campo vengono usate indiscriminatamente lettere maiuscole e minuscole.

Per il computer, le minuscole seguono le maiuscole, cosicché, tanto per fare un esempio, "ABC Cinema" precederà "Abate S.p.A.".

Conviene, dunque, uniformare il formato dei dati immessi.

Durante la visione dei record, ci capiterà di osservare:

A(VANTI) I(NDIETRO) M(ENU)

scritto sul fondo dello schermo.

Premendo adesso il tasto A, viene mostrato il record successivo (una ripetuta pressione del tasto permette di scorrere l'intero file).

Una pressione del tasto I, consente la visione del record precedente, cosicché, grazie ai due tasti A e I, possiamo esaminare l'intero file.

Il tasto M ci riporta invece al menu principale.

Al di sotto della scritta A(VANTI) I(NDIETRO) M(ENU), compare quest'altra:

V(ARIAZIONE) C(ANCELLAMENTO) S(TAMPA)

Queste funzioni sono spiegate più avanti, alle pagine 75-79.

SAVE E LOAD

Come si vedrà, il menu principale contiene le opzioni 5 e 6 (rispettivamente, SCRITTURA e LETTURA FILE). Con l'eccezione della versione Spectrum, infatti, i dati vengono memorizzati separatamente dal programma.

Desiderando archiviare il nostro file, premiamo il tasto [5] e il computer chiederà quale nome usare, digitato il quale (seguito da [ENTER] o da [RETURN]), compare la scritta 'SCRITTURA DEI DATI'.

Sul Commodore, il Dragon e gli Acorn vengono salvati i soli dati.

Volendo memorizzare il programma, occorre selezionare l'opzione 7 (FINE LAVORO) e procedere con le operazioni specificatamente previste per ogni particolare computer.

Quando, in un momento successivo, riutilizzeremo il programma (sul Commodore, il Dragon e gli Acorn), l'operazione consisterà di due fasi.

La prima è il caricamento del programma, da effettuarsi secondo le normali procedure.

Quindi, lanciato il programma, si seleziona l'opzione 6 "LETTURA FILE". Viene chiesto il nome dell'archivio da esaminare: lo si digiti (seguito da [RETURN] o [ENTER]) e il computer visualizzerà il messaggio "AVVIARE IL NASTRO E PREMERE QUALSIASI TASTO".

Il computer cercherà il file sul nastro, e, una volta trovato, lo caricherà in memoria. In tal caso, apparirà il messaggio "LETTURA COMPLETATA". Se la ricerca ha esito negativo, vengono visualizzati i nomi incontrati. A questo punto, mediante l'opzione 6 del menu, si selezioni il nome di un altro file.

Sullo Spectrum, i dati vengono memorizzati assieme al programma, ma una volta che il programma sia stato caricato secondo le procedure usuali, è possibile usare l'opzione 6 per passare a un altro archivio.

VARIAZIONI E CANCELLAZIONI

Le versioni del programma, qui riportate, presentano alcune vistose lacune nel numero delle linee (come si può osservare, ad esempio, pressappoco dalla linea 2000 alla linea 3000).

Si trascrivano usando la numerazione indicata, perché tali lacune di linee verranno successivamente riempite con le parti che riguardano la variazione e la cancellazione dei record e per la ricerca incrociata. I particolari di queste funzioni vengono trattati nella prossima lezione di questa sezione.

```
2 princhr$(8):gosub6:goto100
6 dimfc(7,1):fori=1to7:readfc(i,0):fc(i,1)=
    -1:next:data-1,,,,,-1,-1
12 dimof(3):of(0)=64:of(1)=0:of(2)=
    128: of (3)=64
14 dimlx(8),hx(8)
20 vic=0
22 bd=53280:bg=53281:bb=0
24 cc$=chr$(5):bc=0
28 sb=1024:ll=40:sh=25
40 pokebg,0:pokebd,0
42 printchr$(14)
44 gr$=chr$(30):pu$=chr$(156):yl$=
chr$(158)
46 cs$=chr$(147):ch$=chr$(19)
48 cd$=chr$(17):cu$=chr$(145)
50 rv$=chr$(18):ro$=chr$(146)
52 cl$=chr$(157):cr$=chr$(29):c4$=cd$
    +cd$+cd$+cd$
56 dl$=chr$(20):d4$=dl$+dl$+dl$+dl$:
    is$=chr$(148)
58 rt$=chr$(13)
60 qt$=chr$(34):cm$=chr$(44)
70 ul$=right$("————————————
    ————————————",ll)
72 x1$=cd$+rv$+cc$:x2$=ro$+"□"+
    gr$+rv$:x3$=ro$+"□□□"+rv$+
    cc$
74 x4$=x4$+cu$
76 dimm$(7):fori=1to7:readm$(i):next
78 data"□Creazione di un file□□□□",
    "Immissione record□□","□Visionerecord
    □□□"
80 data"□Ricerca record□□□","□Scrittura
    file□□","□Lettura file□□"
82 data"□Fine lavoro□"
84 dimau$(6):fori=1to6:readau$(i):next
86 data"Avanti","□Indietro□□",
```

```
    "□Menu□□","□Variazione□",
    "Cancellamento□","□Stampa□"
96 w1$=rv$+pu$+"□SEI SICURO
    (s/n)□?□"+cc$+ro$:return
100 printcs$+cc$+ul$
110 printpu$"□*□□□*□□*□□*□*□□*
    □□□*□□□*□***□*□□*□*□□*"
120 print"□**□**□*□*□*□**□*□□□**□
    **□*□□□**□*□*□□*"
130 print"□*□*□*□***□*□*□**□□□*□*
    □*□**□□*□**□*□□*"
140 print"□*□□□*□*□*□*□*□□*□□□
    *□□□*□*□□□*□□*□*□□*"
150 print"□*□□□*□*□*□*□*□□*□□□
    *□□□*□***□*□□*□□**"
160 printcc$ul$
500 print 1$"1"x2$m$(1)x3$;
510 iffd=0then 550
520 print"4"x2$m$(4)
530 printx1$"2"x2$m$(2)x3$"5"x2$m$(5)
540 printx1$"3"x2$m$(3)x3$;
550 print"6"x2$m$(6)
560 printtab(11)x1$"7"x2$m$(7)c4$
600 printtab(11)x4$
    "□QUALE□□ OPZIONE□"ro$:ford=
    1to250:next
610 geta$:ifa$=""then650
620 a=asc(a$)-48:ifa<1 ora>7thengosub
    10000:goto650
630 goto 700
650 printtab(11)x4$rv$"□QUALE□□
    OPZIONE□":ford=1to250:next
660 geta$:ifa$=""then 600
670 a=asc(a$)-48:ifa<1 ora>7thengosub
    10000:goto600
700 ifnotfc(a,-fd)thengosub10000:goto600
800 iffc(a,0)=0orfd=0then890
810 ix$=m$(a):gosub14500
820 ifaa$<>"s"then 100
830 ifa=7then900
840 poke631,a:clr:gosub6:a=peek(631):goto
    700
890 ifu=0and((a=3)or(a=4))thengosub
    10000:goto600
900 onagosub1000,2000,3980,4000,950,950,
    7000
910 goto100
950 printro$cs$gr$tab(11)m$(a)c4$
960 print"Nome del file?":x=16:y=5-vic:z
    =10:gosub13000:f$=ix$
972 iff$=""then950
980 ifa=6then6000
990 ifa=5then5000
1000 printcs$cd$"Numero di campi (1-8):?□";
1010 ok$="12345678":gosub10600:nf=ix:
    printaa$cd$:y=3-vic:tt=5-sh-2*vic
1020 forn=1 tonf
1030 print"Testata"n"□:?□";
1040 x=20:z=10:gosub13000
1045 iflen(ix$)=0thenprintro$rt$cu$cu$:goto
    1030
1050 hd$(n)=ix$
1060 printtab(13):printd4$d4$d4$:x=0:gosub
    11500
1070 print"Lunghezza del campo□"ix$"□?"
1080 x=23+len(hd$(n)):z=2:gosub13000:
    ifix$=""thengosub10000:goto1080
1090 gosub12000:hx(n)=2+int((12+ix)/ll)
1092 ifix<1 orhx(n)>3thengosub10000:goto
    1080
1094 tt=tt+hx(n)
1096 lx(n)=ix:y=y+2:print:print
1098 next
1100 ln=0:fori=1 tonf:ln=ln+lx(i):next:fr
    =fre(0):iffr<0thenfr=fr+65536
1110 v(int(fr/(ln+5+3*nf))
1120 print"□record disponibili": ford
    =1to1500:next
1130 dimt$(v,nf-1),r(v)
1140 u0=0
1200 foru=u0tov
1210 printcs$cc$rv$"Record usati:"u"██
    □dei "v"██□disponibili"cd$
1220 up=u:r(up)=up
1230 forix=1 tonf
1240 gosub3720
1250 ifix=1 andix$=""then1400
1260 fori=1to500:next
1280 next
1300 ifu=0then1340
1302 ix$=t$(u,0):ru=u:su=u
1310 foru2=0tou-1
1320 ift$(r(u2),0)>ix$then1350
1330 next
1340 u2=su:goto1380
1350 fordn=utou2+1step-1
1360 r(dn)=r(dn-1)
1370 next
1380 r(u2)=ru:ifa>2thenup=u2:printch$cc$
    rv$"□RECORD N.□"up+1:goto330
1390 nextu
1400 fd=-1
1990 return
2000 u0=u:b=1
2100 goto1200
3000 u0=up-1
3010 ifu0<0thenu0=u-1:ifa=4then3920
3020 forup=u0thu-1
3030 ifa=4then4110
3040 printcs$cc$rv$"□RECORD N.□"up+1
3050 forix=1 tonf:gosub3770:next
3100 x=0:y=sh-2+2*vic:gosub11500
3110 fori=1 to6:printx3$x2$au$(i);: ifi=3
    thenprintro$"□□□□□□□";
3120 next
3200 ok$="aimvcs□":gosub10600
3210 b=ix
3300 printro$;:onbgoto3900,3000,1990,3700,
    3400,3600,3900
3400 goto100
3600 gosub3720:print"Stampante pronta
    (s/n)?";:gosub10500
3610 ifaa$="n"thengosub3720:goto3100
3620 open4,4:cmd4
3630 print#4,"□con il record□"up+1
    "□sono usati□"u"□record"
3640 forn=1 tonf:print#4:print#4,"□"
    hd$(n)":"spc(12-len(hd$(n)))t$(r(up),n
    -1)
3650 next:print#4,ul$:close4:goto3100
3700 goto100
3720 y=sh-2+2*vic:x=0:gosub11500:
    printro$;:z=ll*(2-2*vic)-2:gosub13500
3722 ifb=6thenreturn
3730 printhd$(ix)ro$"□:";:x=12:z=lx(ix):
    gosub11500
3760 gosub13010:t$(r(up),ix-1)=ix$
3770 y=2-vic:ifix>1thenforn=1 toix-1:
    y=y+hx(n)+(n<=tt):next
3780 x=0:gosub11500:printro$hd$(ix)"□:"
    tab(13);:ifa>2thengosub13500
3790 printt$(r(up),ix-1):return
3800 ifix>1then3100
3810 ifup>0thenift$(r(up),0)<t$(r(up-1),0)
    then3830
3820 ifup=u-1ort$(r(up),0)<=t$(r(up
    +1),0)then3100
3830 ix$=t$(r(up),0):ru=r(up):ifup=u-1
    then3850
3840 fordn=uptou-1:r(dn)=r(dn+1):next
3850 su=u-1
3855 goto 1310
3900 nextup
3910 ifa<>4then3980
3920 printcs$x1$"□FINE DEL FILE
    DURANTE LA RICERCA"
```

Microtip

La trascrizione dei programmi lunghi

Trascrivere lunghi programmi, specie se scritti da altri, può essere un compito laborioso, forse scoraggiante. Per alleggerirlo, si può copiare una breve sezione alla volta, per poi confrontarla con l'originale. Alcuni programmi sono già suddivisi in più sezioni o sottoprogrammi, altrimenti si trascrivano 20 o 30 righe alla volta. Fortunatamente, più si impara a programmare, meglio si comprende ciascuna linea e meno errori verranno commessi.

```
3930 printx1$
  "□DEVO RIPARTIRE DALL'INIZIO (s/n)?":
  gosub10500
3935 ifaa$="s"thengosub4000
3940 ifaa$<>"s"thenreturn
3950 ifb=2then3020
3980 u0=0:b=1
3990 goto3010
4000 return
4110 ix$=t$(r(up),fx-1)
4120 fe=len(ix$)-ff+1
4130 iffe<1 then4160
4140 forj=1 tofe:ifmid$(ix$,j,ff)=fx$then
  3040
4150 next
4160 ifb=2then3000
4170 goto3900
5000 printcs$rv$"□□□□□□□□
  □POSIZIONE NASTRO PER LA SCRITTURA
  □□□□"
5005 print"□□□□□□ Premere il tasto
  return appena pronti."
5010 geta$:ifa$<>rt$then5010
5100 open 1,1, 1,f$
5110 print"□Fase di scrittura□"
5120 print#1,u;cm$nf;cm$tt
5130 forn=1 tonf:print#1,qt$hd$(n)qt$cm$
  lx(n)cm$hx(n):next
5140 forup=0tou:forn=1 tonf:print#1,qt$t$
  (up,n-1)qt$:next:print#1,r(up):next
5150 close1
5990 return
6000 printcs$rv$"□□□□□□□□□
  POSIZIONARE NASTRO PER LA
  SCRITTURA □□□□□"
6005 print"□□□□□□ Premere il tasto
  return appena pronti."
6010 getaa$:ifaa$<>rt$then6010
6100 open 1,1,0,f$
6110 print"File trovato: lettura"
6120 input#1,u,nf,tt
6130 forn=1 tonf:input#1,hd$(n),lx(n),hx(n):
  next
6140 ln=0:forn=1 tonf:ln=ln+lx(n):next:fr
  =fre(0):iffr<0thenfr=fr+65536
6150 v=int(fr/(ln+5+3*nf))
6160 dimt$(v,nf-1),r(v)
6200 forup=0tou:forn=1 tonf:input#1,
  t$(up,n-1):next
6210 input#1,r(up):next
6220 close1
6980 fd=-1
6990 return
7000 printcs$:end
10000 poke54277,33:poke54278,255:poke
  54273+23,15
10005 poke54273,6:poke54276,33:ford=1 to
  50:next
10006 poke54273+23,0:poke54273,0
10010 return
10500 ok$="sn"
10600 getaa$:ifaa$=""then10600
10610 ix=0:fori=1 tolen(ok$):ifaa$=
  mid$(ok$,i,1)thenix=i
10620 next:ifix=0thengosub10000:goto
  10600
10630 return
11500 printch$;
11510 ify>0thenforyy=1 toy:printcd$;:next
11520 ifx>0thenforxx=1 tox:printcr$;:next
11530 return
12000 ix=-1:fori=1 tolen(ix$)
12010 a$=mid$(ix$,i,1)
12020 ifa$<>"□"then12050
12030 ifi=1 ori=len(ix$)then12060
12040 ifmid$(ix$,i-1,1)="□"then12060
12050 ifa$<"0"ora$>"9"thengosub10000:
  return
12060 next
12070 ix=val(ix$):return
13000 gosub11500:gosub13500
13010 p0=sb+ll*y+x:p1=p0:i=128:ix$
  =""
13020 pokep1,(peek(p1)and127)or(iand128)
13030 geta$:i=(i+12)and255:ifa$=""then
  13020
13040 ifa$=dl$then13150
13050 ifa$=rt$thenreturn
13060 ifasc(a$)and127<32then13190
13100 if p1>= p0+z then 13190
13110 pokep1,peek(p1)and127:p1=p1+p1:
  printa$;:ix$=ix$+a$:goto13020
13150 ifp1=p0then13190
13160 pokep1,peek(p1)and127:p1=p1-1:
  printcl$"□"cl$;:ix$=left$(ix$,p1-p0)
13170 goto13020
13190 gosub10000:goto13020
13500 fori=1 toz:print"□";:next
13510 fori=1 toz: printcl$;:next
13520 return
14500 b=6:gosub3720:printcd$"□□□□□
  □□□"pu$w1$yl$rv$cu$
  "█▌█▌█▌█▌█▌█▌█▌█▌█▌█▌█▌█▌█▌
  █▌█▌█▌█▌█▌█▌█▌█▌  □"ix$;
14530 gosub10500
14540 return
```

```
5 LET R=0: LET U=0: LET V=1
10 BORDER V: PAPER V: INK 7: POKE 23609,
  20: POKE 23658,8
100 CLS:PRINT INVERSE V;AT V,6;"□□□□
  □MENU PRINCIPALE□□□□□"
110 PRINT AT 5,6;"1: Creazione file""TAB
  6;"2: Immissione record"" TAB 6;"3:
  Visione record""TAB 6;"4: Ricerca""TAB
  6;"5: Scrittura file""TAB 6;"6: Lettura
  file""TAB 6;"7: Fine lavoro";#V;TAB
  6;"QUALE OPZIONE:"
500 LET I$=INKEY$="" THEN GOTO 500
510 IF I$<"1" OR I$>"7" THEN GOTO 500
520 IF R=U AND I$<>"1" AND I$<
  >"6" AND I$<>"7" THEN GOTO 500
530 BEEP .1,10: CLS: GOSUB (CODE I$
  -48)*1000: GOTO 100
1000 PRINT AT 7,9;"SEI SICURO (S/N)?":
  PAUSE U: IF INKEY$="" THEN GOTO
  1000
1010 IF INKEY$<>"S" THEN RETURN
1020 PRINT INVERSE V;AT
  10,6;"□CREAZIONE NUOVO FILE□"
1030 INPUT AT 0,0;"Numero di campi (1-
  8)?□";A: IF A<1 OR A>8 THEN GOTO
  1030
1040 DIM A(A): DIM B(A+V): DIM
  N$(A,10):LET T=U: FOR N=V TO A
1050 INPUT AT 0,0;"Nome del
  campo□";(N);"□?□"; LINE N$(N)
1060 INPUT AT V,0;"Lunghezza del
  campo□";(N);"□?□";A(N): IF A(N)>50
  THEN GOTO 1060
1070 LET B(N)=T: LET T=T+A(N):
  NEXT N:LET B(N)=T
1080 PRINT AT 16,2;"Spazio per
  circa□";INT(((PEEK 23730+256*PEEK
  23731)-29500)/T); "□record"
1090 INPUT "Quanti sono da usare?□";R:
  DIMA$(R,T): RETURN
2000 LET C=V
2010 IF A$(C,V)="□" THEN GOTO 2100
2020 IF C=R THEN GOTO 2500
2030 LET C=C+V: GOTO 2010
2100 PRINT AT 0,0;C
  -V;"□dei□";R;"□record sono in uso"
2110 FOR N=V TO A: PRINT INVERSE V;AT
  V+N*2,U;N$(N); INVERSE 0;AT V
  +N*2,12; FLASH V;"?": INPUT "(fino
  a□";(A(N));"□caratteri)", LINE A$(C,B(N)
  +V TO B(N+V)): PRINT AT V
  +2*N,12;A$(C,B(N)+V TO B(N+V)):
  NEXT N
2120 FOR F=V TO 150: NEXT F: IF C=V
  THEN RETURN
2130 LET N=C
2140 IF A$(C)>=A$(C-V) THEN RETURN
2150 LET X$=A$(C): LET A$(C)=A$(C-V):
  LET A$(C-V)=X$: LET C=C-V: IF C
  =V THEN RETURN
2160 GOTO 2140
2500 CLS:PRINT FLASH 1;AT 10,6;"□F□I□
  L□E□□□P□I□E□N□O□":FOR F
  =V TO 400:NEXT F:RETURN
3000LET D=V: IF A$(V,V)="□" THEN
  RETURN
3010 IF D=U THEN LET D=V
3015 IF D-V=R THEN LET D=D-V
3020 IF A$(D,V)="□" THEN LET D=D-V
3030 GOSUB 9500
3040 IF OP=V THEN LET D=D+V: GOTO
  3010
3050 IF OP=2 THEN LET D=D-V: GOTO
  3010
3060 IF OP=3 THEN RETURN
3070 IF OP=4 THEN GOSUB 8000
```

```
3080 IF OP=5 THEN LET MD=V: GOSUB
    9000:IF D=U THEN RETURN
3090 GOTO 3030
4000 REM LINEA PROVVISORIA
5000 INPUT "Nome del file□□"; LINE Q$:IF
    LEN Q$<V OR LEN Q$>10 THEN GOTO
    5000
5010 SAVE Q$ LINE 10: RETURN
6000 PRINT AT 8,U;"Nome del file da leggere,
    o solo ENTER per il primo file"
6010 INPUT LINE X$: IF LEN X$>10 THEN
    GOTO 6010
6020 PRINT AT 13,U;"AVVIARE IL
    REGISTRATORE": LOAD X$
7000 PRINT "Sei sicuro (S/N)?":IF INKEY$
    ="" THEN GOTO 7000
7010 IF INKEY$<>"S" THEN RETURN
7020 RANDOMIZE USR U
8000 RETURN: REM LINEA PROVVISORIA
9000 RETURN: REM LINEA PROVVISORIA
9500 PRINT AT U,U;"N. del record:□";D;
    "□□": FOR N=
    V TO A: PRINT INVERSE V;AT V+2*N,U;
    N$(N); INVERSE U;TAB 12;A$(D,B(N)+
    V TO B(N+V)): NEXT N
9510 PRINT INVERSE V;AT 20,U;"□A(vanti)
    □□I(ndietro)□□□□M(enu)□□□□□
    V(ariazione)□□□□C(ancellamento)□□
    S(tampa)□"
9520 IF INKEY$="" THEN GOTO 9520
9530 LET V$=INKEY$: IF V$="S" THEN
    COPY:LPRINT: LPRINT: LPRINT: GOTO
    9520
9540 LET OP=U: IF V$="A" THEN LET OP
    =V:LET MO=V
9550 IF V$="I" THEN LET OP=2: LET MO
    =-V
9560 IF V$="M" THEN LET OP=3
9570 IF V$="V" THEN LET OP=4
9580 IF V$="C" THEN LET OP=5
9590 IF OP=U THEN GOTO 9520
9600 BEEP .1,10: RETURN
```

T

```
20 PCLEAR1:CLEAR 11000:RS$="A□
    IMVCS":B$=CHR$(128)
30 CLS:PRINT@39,B$;"m";B$;"e";B$;"n";B$;
    "u";B$;B$;B$;"p";B$;"r";B$;"i";B$;"n";B$;
    "c";B$;"i";B$;"p";B$;"a";B$;"l";B$;"e";B$
40 PRINT@164,"1:—CREAZIONE DEL FILE"
50 PRINT @196,"2:— IMMISSIONE RECORD"
60 PRINT@228,"3:— VISIONE RECORD"
70 PRINT@260,"4:— RICERCA"
80 PRINT@292,"5:— SCRITTURA DEL FILE"
90 PRINT@324,"6:— LETTURA DEL FILE"
100 PRINT@356,"7:— FINE LAVORO"
110 PRINT@481,"QUALE OPZIONE?";
120 IN$=INKEY$:IFIN$<"1"ORIN$>"7"
    THEN120
130 IFIN$<>"1" ANDIN$<>"6"ANDR
    =0 ANDIN$<>"7"THEN120
140 SOUND30,1:CLS:IN=VAL(IN$)
150 ON IN GOSUB1000,2000,6000,5000,
    7000,8000,9000
160 GOTO30
1000 PRINT@41,"CREAZIONE DEL FILE":
    PRINT@231,"SEI SICURO (S/N)?"
1010 IN$=INKEY$:IFIN$<>"S"AND IN$<
    >"N" THEN1010
1020 IFIN$<>"S"THENRETURN
1030 IFR>0 THENRUN9200
1040 CLS:PRINT@41,"CREAZIONE DEL FILE"
1050 PRINT@385,"NUMERO DI CAMPI (1-
    8)";:INPUTA:A=ABS(INT(A))
1060 IFA>8 ORA<1 THEN1050
1070 DIM A(A),NS(A)
1080 PRINT@384," ":PRINT@96," ":FORN
    =1TOA
1090 PRINT:PRINT"NOME DEL CAMPO";
    N;"?";:LINEINPUTN$(N):N$(N)=LEFT$
    (N$(N),10)
1100 PRINT"LUNGHEZZA DEL CAMPO";
    N;:INPUTA(N):A(N)=ABS(INT(A(N)))
1110 IFA(N)>19 OR A(N)<1 THEN1100
1120 TS=TS+A(N)
1130 NEXT:R=INT(11000/(5+5*A))-1:
    PRINT"MASSIMO NUMERO DI RECORD
    =";R
```

```
1140 DIMA$(R,A):FORI=1TO2000:NEXT:
   RETURN
2000 G=0
2010 IFNR=R THEN2180
2020 NR=NR+1
2030 CLS:PRINT@0,NR-1;"□DEI□";R;"□
   RECORD SONO IN USO"
2040 FORN=1TOA:PRINT@32*N+32,N$(N);
   "□:":PRINT@448," ":PRINT@416," "
2050 PRINT@416,"(FINO A";A(N);
   "CARATTERI)□□";:LINEINPUTA$(NR,N)
2060 IFA$(NR,N)="" AND N=1THENN=A:
   G=1:GOTO2080
2070 A$(NR,N)=LEFT$(A$(NR,N),A(N)):
   PRINT@32*N+45,A$(NR,N)
2080 NEXT
2090 IFG=1 THEN 2160
2100 C=NR:FORF=1TO150:NEXT:IF
   NR=1 THEN2150
2110 IFA$(C,1)>=A$(C-1) THEN2150
2120 FORN=1TOA:X$=A$(C,N):A$(C,N)=
   A$(C-1,N):A$(C-1,N)=X$: NEXT: C
   =C-1
2130 IFC=1 THEN2150
2140 GOTO2110
2150 GOTO2010
2160 NR=NR-1
2170 RETURN
2180 CLS3:PRINT@235,"□FILE PIENO□";:
   FORG=1TO5:SCREEN0,1:FORF=1TO
   500: NEXT
2190 SCREEN0,0:FORF=1TO500:NEXTF,G:
   RETURN
3000 RETURN: REM LINEA PROVVISORIA
4000 RETURN: REM LINEA PROVVISORIA
5000 RETURN: REM LINEA PROVVISORIA
6000 D=1
6010 IFNR<1 THEN6170
6020 GOSUB8500
6030 PRINT@451,"aVANTI□□□□iNDIETRO
   □□□mENU□□vARIAZIONE□□□□□
   cANCELLAMENTO□□□□□sTAMPA";
6040 IN$=INKEY$:IFIN$="" THEN6040
6050 IN=INSTR(1,RS$,IN$)
6060 ON IN GOTO 6080,6080,6090,6100,
   6110,6120,6130
6070 GOTO6030
6080 D=D+1:GOTO6140
6090 D=D-1:GOTO6140
6100 RETURN
6110 GOSUB3000:GOTO6020
6120 GOSUB4000:GOTO6010
6130 GOSUB10000:GOTO6030
6140 IFD>NR THEND=1
6150 IFD<1 THEND=NR
6160 GOTO6010
6170 CLS3:PRINT@233,"□FILE VUOTO□";
6180 FORG=1 TO5:SCREEN0,1:FORF=1 TO
   300:NEXT:SCREEN0,0:FORF=1 TO300:
   NEXTF,G:RETURN
7000 AUDIOON:MOTORON:CLS:PRINT@65,
   "PREPARARE IL REGISTRATORE
   POSIZIONARE IL NASTRO E PREMERE
   ENTER";
7010 IN$=INKEY$:IFIN$<>CHR$
   (13) THEN7010
7020 MOTOROFF:PRINT@129,"PREPARARE
   IL REGISTRATORE PER LA SCRITTURA E
   PREMERE ENTER";
7030 IN$=INKEY$:IFIN$<>
   CHR$(13) THEN7030
7040 PRINT:INPUT"□NOME DEL FILE□";FI$
7050 CLS6:PRINT@232,"SCRITTURA□";FI$;
7060 MOTORON:FORI=1TO1000:NEXT
7070 OPEN"O",#-1,FI$
7080 PRINT#-1,FI$,R,A,NR
7090 FORN=1TOA:PRINT#-1,NS(N),A(N):
   NEXT
7100 C=1
7110 IFA$(C,1)="" THEN7140
7120 FORN=1TOA:PRINT#-1,A$(C,N):
   NEXT
7130 C=C+1:GOTO7110
7140 CLOSE#-1:RETURN
8000 CLS:PRINT@70,"SEI SICURO (S/N)?"
8010 IN$=INKEY$:IFIN$<>"S"ANDIN$<
   >"N" THEN8010
8020 IFIN$="N" THENRETURN
8030 AUDIOON:MOTORON:CLS:PRINT@65,
   "POSIZIONARE IL NASTRO E PREMERE
   ENTER"
8040 IN$=INKEY$:IFIN$<>
   CHR$(13) THEN8040
8050 MOTOROFF:PRINT@129,"PREPARARE
   IL REGISTRATORE PER
   LETTURA□□□E PREMERE ENTER"
8060 IN$=INKEY$:IFIN$<>
   CHR$(13) THEN8060
8070 IFR>0 THENRUN9210
8080 INPUT"NOME DEL FILE";FI$
8090 CLS7:PRINT@231,"RICERCA□";
8100 OPEN"I",-1,FI$
8110 INPUT#-1,FI$
8120 PRINT@231,"□TROVATO□□";
   FI$;"□";
8130 INPUT#-1,R,A,NR
8140 DIMA(A),N$(A),A$(R,A)
8150 FORN=1TOA:INPUT#-1,N$(N),A(N):
   NEXT
8160 C=1
8170 IFEOF(-1) THEN8200
8180 FORN=1TOA:INPUT#-1,A$(C,N)
8190 NEXT:C=C+1:GOTO8170
8200 CLOSE#-1:RETURN
8500 CLS:PRINT@0,"NUMERO RECORD";D:
   FORN=1TOA:PRINT@32*N+32,N$(N);
   "□:";TAB(13);A$(D,N):NEXT:RETURN
9000 CLS4:PRINT@70,"SEI SICURO (S/N)?";
9010 IN$=INKEY$:IFIN$<>"S"ANDIN$<
   >"N" THEN9010
9020 IFIN$="N" THENRETURN
9030 CLS:END
9200 GOSUB1040:GOTO9220
9210 GOSUB8080
9220 B$=CHR$(128):RS$="A□IMVCS":
   GOTO30
10000 PRINT@451,□□□STAMPANTE□OK
   □□□□□□□□cONT□";
10010 PRINT@480,"□□□□□□□□
   □□□□□□□□□□□□□□□□
   □□□□□□□□";
10020 IN$=INKEY$:IFIN$="" THEN10020
10030 IFIN$<>"C" THENRETURN
10040 FORY=0TOA+4:FORX=0TO31:P
   =PEEK(1024+X+Y*32):IFP>95ANDP
   <127 THENP=P-64
10050 IFP>0ANDP<27 THEN=P+96
10060 IFP=0 THENP=32
10070 PRINT#-2,CHR$(P);:NEXT:PRINT#
   -2,CHR$(13);:NEXT
10080 FORN=1TO3:PRINT#-2,CHR$(13):
   NEXT
10090 RETURN
```

```
1MODE7:M%=0:N%=1
2ONERRORGOTO13000
3*OPT1,1
4HIMEM=8999
5DIMA(8),N$(8),TRL(8)
6VDU23;8202;1;0;0;0;0;
7B%=HIMEM+1
8*OPT3,6
30CLS:PRINT''"□□□□□□
   MENU PRINCIPALE"
40 PRINT''□□□□□□□1:
   -Creazione del file"
50 PRINT''"□□□□□□□2:
   -Immissione record"
60 PRINT'"□□□□□□□3: -Ricerca"
70 PRINT'"□□□□□□□4: -Scrittura del file"
80 PRINT'"□□□□□□□5: -Lettura del file"
90 PRINT'"□□□□□□□6: -Fine lavoro"
100 PRINT''□□□□□□□Opzione scelta"
130 G=GET-48:IF G<1 OR G>7 THEN
   130
140 IF M%=0 AND (G>1 AND G<6)
   THEN130
145 IF G<>7 THEN 143
146 CLS:PRINTTAB(13,12)"Sei sicuro?":G
   =GET AND &5F
147 IF G=83 THEN END ELSE 30
148 ON G□GOTO 150,160,170,180,190,200
150 PROCNEWFILE:GOTO 30
160 PROCENTER:GOTO 30
170 PROCVIEW:GOTO 30
180 PROCSEARCH:GOTO 30
190 PROCSAVE:GOTO 30
200 PROCLOAD:GOTO30
1000 DEF PROCNEWFILE
```

```
1005 CLS:PRINT'"Sei sicuro (S/N)?"
1010 G=GET AND &5F:IF G<>83 THEN
   1110
1015 N%=1:R%=0
1020 CLS:PRINT"Creazione nuovo file"
1040 PRINT'"Numero di campi (da 1 a 8)?"
1050 A=GET-48:IF A<1 OR A>8 THEN
   1050
1058 FOR N=1 TO A
1059 PRINT'"Nome del campo□";N;"□":
   PROCINPUT(10):IF $B%=" "THENPROC
   INPUT(10)
1060 N$(N)=$B%
1065 PRINT'"Lunghezza massima del
   campo□";N;"□";
1075 INPUTA(N):IF A(N)>27 OR A(N)<1
   THEN 1075
1078 TRL(N)=R%:R%=A(N)+1+R%
1090 NEXT:IFR%<11 THEN R%=11
1100 M%=INT ((&7C00-HIMEM)/R%):
   PRINT" "Spazio per□";M%;"□record":D%
   =INKEY(300)
1110 ENDPROC
2000 DEF PROCENTER
2002 IF N%=M%+1 THEN CLS:PRINT'
   "File pieno":G=INKEY(300):GOTO2130
2003 Q=0:CLS:PRINTN%-1;"□dei□";M%;
   "□record sono in uso"
2007 PRINT'"Premere RETURN per il MENU
   oppure si□□continui l'immissione"
2010 FOR N=1TOA:PRINTTAB(0,3+N*2)
   N$(N);"□:"
2020 PRINTTAB(0,23)STRING$(27,"□")
   TAB(0,23);:PROCINPUT(A(N))
2021 PRINTTAB(13,3+N*2);$B%
2022 IF ASC($B%)=-1 AND N=1 THEN N
   =A:Q=1:GOTO 2030
```

Modifiche per l'Electron

Per l'Electron, occorre apportare qualche modifica al programma. Non esistendo il Mode 7, si adoperi il Mode 6. La linea 1 diventa:

```
1 MODE6:M%=0:N%=1
```

Si aggiungano quindi le linee:

```
9004 IF G=67 THEN VDU2
9025 IF G=67 THEN VDU3
```

Sostituire il valore &7C00, alla linea 1100, con &6000.
Sostituire la linea 9500 con:

```
9500 DEF PROCSTAMPANTE
```

e si aggiungano:

```
9503 PRINT "Stampante ok? - C(conti-
   nua)":G=GET AND &5F: IF G<>67
   THEN 9540
9510 PROCVDU
```

```
2025 $(B%+N%*R%+TRL(N))=$B%
2030 NEXT
2035 IF Q=1 THEN 2130
2037 FOR T=1 TO 2000:NEXT
2040 N%=N%+1
2070 IF N%=2 THEN 2000
2080 G=N%-1
2090 X=B%+G*R%:Y=B%+(G-1)*R%:IF
   $X>=$Y□THEN 2000
2100 FOR T=1 TO A:$B%=$(X+TRL(T)):$
   (X+TRL(T))=$(Y+TRL(T)):$(Y+TRL(T))
   =$B%:NEXT
2110 G=G-1: IF G=1 THEN 2000
2120 GOTO2090
2130 ENDPROC
3000 DEF PROCAMEND
3190 ENDPROC
4000 DEF PROCDELETE
4090 ENDPROC
5000 DEF PROCSEARCH
5100 ENDPROC
6000 DEF PROCVIEW
6002 IF N%=1 THEN ENDPROC
6005D%=0:C=1:Q=0
6010 REPEAT
6020 D%=D%+C
6023 IF D%>N%-1 THEN D%=1
6025 IF D%<1 THEN D%=N%-1
6030 PROCVDU
6035 PROCKEY
60371 F N%<2 THEN Q=1
6040 UNTIL Q=1
6050 ENDPROC
7000 DEF PROCSAVE
7002 CLS
7003 PRINT"Nome del file?□□
   □□□□□□□□□□";TAB(14,0);:PROC
   INPUT(10)
7004 PRINT"
7010 IF LEN($B%)<1 THEN 7003
7030 X=OPENOUT $B%
7035 PRINT'"Fase di scrittura"
7040 PRINT#X,M%,N%,A,R%
7050 FOR N=1 TO A:PRINT#X,N$(N),A(N),
   TRL(N):NEXT
7055 Y=B%+R%:Z=B%+R%*N%
7060 FOR T=Y□TO Z
7070 BPUT#X,?T
7080 NEXT
7090 CLOSE#X
7100 ENDPROC
8000 DEF PROCLOAD
8002 CLS
8003 PRINT"Nome del file?□□□□
   □□□□□□□□";TAB(18,0);:PROC
   INPUT(10)
8004 PRINT'"AVVIARE IL REGISTRATORE"
8007 X=OPENIN $B%
8010 INPUT#X,M%,N%,A,R%
8015 B%=HIMEM+1
8020 FOR N=1 TO A:INPUT#X,N$(N),A(N),
   TRL(N):NEXT
8025 Y=B%+R%:Z=B%+R%*N%
8030 FOR T=Y□TO Z
8040 ?T=BGET#X
8050 NEXT
8060 CLOSE#X
8065 VDU13:PRINT'"LETTURA CONCLUSA":
   G$=INKEY$(300)
8100 ENDPROC
9000 DEFPROCVDU
9003 CLS
9005 PRINT"Numero record□";D%'
9010 FOR S=1 TO A:PRINTN$(S);"□:"
   TAB(13);$(B%+D%*R%+TRL(S))
9020 NEXT
9030 PRINT'"A(vanti)□□□□□□I(ndietro)
   □□□□□□□□M(enu)"'"V(ariazione)□
   □□□□□C(ancellamento)□□□□□□
   S(tampa)"
9035 ENDPROC
9037 DEF PROCKEY
9040 G=GET AND &5F
9024 IF G=65 OR G=0 THEN C=1:GOTO
   9100
9044 IF G=77 THEN Q=1:GOTO 9100
9045 IF G=83 THEN PROCPRINTER:GOTO
   9040
9047 IF G=86 THEN PROCAMEND:GOTO
   9100
9050 IF G=73 THEN C=-1:GOTO 9100
9055 IF G=67 THEN PROCDELETE:GOTO
   9100
9060 GOTO 9040
9100 ENDPROC
9500 DEF PROCPRINTER:PRINT'
   "Stampante ok-C(continua)":G
   =GET AND &5F:IF G<
   >67 THEN 9540 ELSE VDU2:FOR Y
   =0 TO A*2+3:FOR X=0 TO 39:VDU1,
   ?(&7C00+Y*40+X):NEXT:VDU13:NEXT:
   VDU3
9540 VDU11:PRINTSTRING$ (40,"□"):VDU11,
   11,11
9550 ENDPROC
12000 DEF PROCINPUT(X)
12010 $B%=" ":FOR T=1 TO X+1
12020 K=GET
12030 IF K=127 AND T>1 THEN T=T
   -1:VDU 127:$B%=LEFT$($B%,T-1):
   GOTO12020
12050 IF K=13 THEN T=X+1:
   GOTO12100
12060 IF K<>13 AND T=X+1 THEN
   12020
12070 IF K<32 OR K>126 THEN12020
12080 $B%=$B%+CHR$(K)
12090 VDU K
12100 NEXT:ENDPROC
13000 IF ERR=17 THEN 30
13010 IF ERR>215 AND ERR<224 THEN
   PRINT'"ERRORE NELLA GESTIONE FILE":
   FOR T=1 TO 7500:NEXT:GOTO30
13050 REPORT: PRINT:END
```

A DESTRA... IN ALTO... A SINISTRA... FUOCO!

Nei giochi tipo 'arcade' occorre avere un controllo completo su quanto avviene sullo schermo. Ecco come gestire efficacemente i movimenti delle varie immagini

Che noia sarebbero i giochi del tipo 'Space Invaders', se non si avesse qualche controllo sui raggi laser o sulla base di lancio! Simili funzioni sono basilari anche nei più banali computer game ed è essenziale comprendere come ottenerli, se vogliamo scrivere dei giochi per conto nostro. La prima mossa consiste nell'ottenere una reazione del computer alla pressione dei tasti.

INDIVIDUARE IL TASTO

Il metodo usato negli home computer per rilevare la pressione dei tasti varia da apparecchio ad apparecchio.

In questi computer, la funzione INKEY$ provoca la scansione della tastiera, per stabilire se e quale tasto sia premuto. Ecco un breve programma di prova:

```
20 CLS
30 IF INKEY$="" THEN GOTO 30
40 PRINT AT 11,14;"AHI!"
```

```
30 LET A$=INKEY$: IF A$="" THEN GOTO
   30
40 PRINT@ 269, "AHI!"
```

Si lanci il programma e si prema un qualsiasi tasto (eccetto i tasti CAPS SHITF e SIMBOL SHIFT sui Sinclair o BREAK o SHIFT sul Dragon e sul Tandy). Al centro dello schermo appare la scritta "AHI!". Ecco una spiegazione del funzionamento.

La linea 20 'ripulisce' lo schermo. La linea 30 provoca un'attesa nell'esecuzione, fino a che non viene premuto un tasto. Si noti che non vi è spazio tra i doppi apici. Ciò vuol dire: se INKEY$ = niente (ossia non viene premuto alcun tasto) allora controlla ancora.

È importante avere IF ... THEN GOTO 30, altrimenti il computer esegue un solo controllo, per una frazione di secondo.

Appena viene premuto un tasto, INKEY$ assume il valore corrispondente al carattere, provocando l'emissione del messaggio "AHI!" sullo schermo (linea 40).

Nella maggior parte dei giochi, i tasti

- L'INDIVIDUAZIONE DEI TASTI
- IL LANCIO DI MISSILI
- IL CONTROLLO DEL MOVIMENTO DELLE IMMAGINI
- LA CREAZIONE DI UN GIOCO TIPO 'ARCADE'
- CACCIA ALL'ALIENO
- BASI MISSILISTICHE
- LA RIPETIZIONE AUTOMATICA
- LE **GET$** E LE **INKEY$**
- TRACCIAMENTO CASUALE DI BERSAGLI

vengono usati per lo spostamento delle immagini. Per meglio comprendere questo fatto, si modifichi la linea 40 (sullo ZX81 usare la D maiuscola):

```
40 IF INKEY$="d" THEN PRINT AT
   11,9;"MMM, CHE BELLO!":STOP
50 PRINT AT 11,14;"AHI!"
```

```
40 IF A$="D" THEN PRINT@ 264,"MMM,
   CHE BELLO!" ELSE PRINT@ 269, "AHI!"
```

La linea 40 controlla se è stato premuto il tasto [D], ossia: INKEY$ contiene D oppure d? Se la riposta è no, lo Spectrum, nel quale non esiste la ELSE, ignora il resto della linea 40 e passa alla 50. (Per capire la funzione di STOP, si provi a ometterla!)

Altre due cose sono importanti in questo programma. La prima è che "D" o "d" devono essere racchiuse tra doppi apici, o il computer le prenderà per variabili. La seconda, solo sullo Spectrum, è che di solito si usa una "d" minuscola, altrimenti occorre premere anche il tasto [SHIFT] per il funzionamento.

Sui computer Acorn si può usare sia GET$ che INKEY$: la differenza principale tra i due è che GET$ sospende l'esecuzione del programma in attesa che venga premuto un tasto, mentre con INKEY$ si può rilevare la pressione di un tasto in qualsiasi momento durante l'esecuzione del programma. Questa è una caratteristica molto importante per i giochi. Ecco un breve programma che usa la GET$:

```
20 CLS
30 tasto$=GET$
40 PRINT TAB(17,13) "AHI!"
```

Si lancia il programma e si preme un qualsiasi tasto alfanumerico.

Il funzionamento è semplice. La linea 30 attende la pressione di un tasto e deposita il carattere corrispondente, che non viene visualizzato, in tasto$.

L'esecuzione passa alla linea 40, con la visualizzazione del messaggio "AHI!" al centro dello schermo.

Il programma risponde con qualsiasi tasto, ma in un gioco si usano diversi tasti per le varie funzioni. Basta modificare la linea 40:

```
40 IF tasto$="D" THEN PRINT TAB(11,12)
   "MMM, CHE BELLO!" ELSE PRINT
   TAB(17,13) "AHI!"
```

La linea 40 controlla se il carattere contenuto in tasto$ è uguale a "D" e, se lo è, viene visualizzato il messaggio "MMM, CHE BELLO!", altrimenti il messaggio "AHI!".

Per individuare la pressione di un tasto, sui Commodore si può utilizzare l'istruzione GET.

Ecco una semplice dimostrazione:

```
20 PRINT "♥"
30 GET A$ : IF A$="" THEN GOTO 30
50 PRINT TAB(17) "AHI!":END
```

Lanciando più volte il programma, si nota che la pressione di qualsiasi tasto (eccetto il [RUN/STOP], lo [SHIFT] e il tasto [Commodore]) lo fa funzionare.

La linea 20 ripulisce lo schermo. La 30, con l'istruzione GET, ha il compito di catturare un carattere. I doppi apici affiancati significano 'nessun tasto premuto' e in tal caso la 'caccia' continua.

La linea 50 visualizza il messaggio "AHI!" sullo schermo, terminando il programma.

Generalmente, serve sapere quale tasto è stato battuto, per selezionare una tra più funzioni diverse (ad esempio il movimento di una navetta o di un raggio laser). Ciò si ottiene aggiungendo una sola linea di programma:

```
40 IF A$="D" THEN PRINT TAB(12) "MMM,
   CHE BELLO!":END
```

La linea 40 viene eseguita ogni volta che si batte un tasto. In essa si controlla che il corrispondente carattere sia una "D" e, in questo caso, viene eseguito il resto della linea.

Altrimenti, l'esecuzione passa alla linea 50. (Perché la END è necessaria? Si provi a ometterla!)

IL LANCIO DI UN MISSILE

Il passo tra l'individuazione della pressione di un tasto, mediante INKEY$ e la creazione di un meccanismo di 'fuoco' per il lancio di un missile è breve. Ecco un programma nel quale, in seguito alla pressione di una [F], si lancia un missile attraverso lo schermo:

```
20 CLS
30 PRINT@ 397,"# # #"
```

```
40 LET K$=INKEY$:IF K$=
   "" THEN GOTO 40
50 IF K$="F" THEN LET M=
   334 ELSE GOTO 40
60 PRINT@ M, "↑"
70 PRINT@ M+32,"□"
80 LET M=M-32
90 IF M>0 THEN GOTO 60
100 GOTO 20
```

La linea 30 crea una 'base di lancio', usando tre graticole (#), a partire dalla posizione 397.

Nella linea 40 l'uso di LET K$=INKEY$ è molto importante, perché si vuole controllare la tastiera più volte nel corso dell'esecuzione e, senza questa istruzione, il programma non funzionerebbe affatto.

Il contenuto di INKEY$ cambia continuamente e, per 'catturarlo', occorre copiarlo in una variabile, K$ per esempio. In questa maniera, il valore 'catturato' può essere controllato anche in seguito.

La linea 50 controlla se il tasto premuto corrisponde alla lettera F. In caso negativo, l'esecuzione torna alla linea 40. M è la posizione del missile *dopo* il lancio.

La linea 60 visualizza il missile, mentre la 70 ripulisce la posizione da esso precedentemente occupata.

La linea 8 sottrae 32 al valore della posizione, cosicché il missile si sposta in verticale sullo schermo. (La ragione per la quale si sottrae 32 è semplice: ci sono 32 colonne su ogni linea dello schermo del Dragon).

La linea 90 impedisce al computer qualsiasi tentativo di visualizzazione esterna allo schermo, il che provocherebbe un errore. Lo schermo inizia alla posizione M=0 e non si può usare la PRINT con un valore di posizione negativo. Quando M=0, il programma riparte automaticamente dall'inizio.

Questo programma lancia un missile premendo il tasto [F]. Sullo ZX81 si usino tutte maiuscole e, alla linea 60, un asterisco (*) al posto della freccia.

```
20 CLS
30 PRINT AT 21,14;"▄█▄"
40 IF INKEY$="" THEN GOTO 40
50 IF INKEY$<>"f" THEN GOTO 40
55 LET y=20
60 PRINT AT y,15;"↑"
70 LET y=y-1
75 PAUSE 1
80 PRINT AT y+1,15;"□"
90 IF y>0 THEN GOTO 60
```

La linea 30 sfrutta un carattere ROM a bassa risoluzione per creare la base di lan-

cio, mentre la linea 4Ø rimane in attesa della pressione di un tasto.

La linea 5Ø controlla che sia stato battuto il tasto [F]. In caso positivo, viene inizializzata la posizione del missile, sulla ventesima riga dello schermo, appena sopra alla base di lancio.

La linea 4Ø provoca una nuova scansione della tastiera, se il carattere non era una F.

La linea 6Ø visualizza il missile. Nello Spectrum, viene usata una freccetta, che si ottiene premendo [SYMBOL SHIFT] e [H].

La linea 8Ø cancella la posizione precedente del missile, mentre la 7Ø sottrae 1 da y, che contiene la riga di visualizzazione, per far muovere verso l'alto il missile.

La linea 9Ø impedisce che il missile esca dallo schermo (quando esso raggiunge il punto più alto dello schermo, y vale Ø).

Il programma lancia un missile alla pressione del tasto [F]:

```
20 CLS
30 PRINT TAB(19,20)"# ∧ #"
40 LET K$=GET$
50 IF K$="F" THEN M=19 ELSE GOTO 40
60 PRINT TAB(20,M)"∧"
70 PRINT TAB(20,M+1)"□"
80 LET M=M-1
90 IF M>Ø THEN GOTO 60
100 GOTO 20
```

La linea 3Ø crea sullo schermo l'immagine della base di lancio.

La linea 4Ø attende la pressione di un tasto e la linea 5Ø controlla se questo corrisponde alla lettera F. In caso affermativo, è assegnata la posizione iniziale al missile, altrimenti l'esecuzione torna alla linea 4Ø. La linea 6Ø visualizza il missile, la 7Ø ne cancella la posizione precedente.

La linea 8Ø sottrae 1 alla posizione del missile, sì da farlo muovere verso l'alto.

La linea 9Ø fa proseguire il moto del missile fino in cima allo schermo, la 1ØØ, infine riavvia il programma.

Questo programma lancia un missile alla pressione del tasto [F]. Sul Vic 2Ø, nella linea 3Ø, sostituire 8 con 18 e eliminare due ■. Nella linea 5Ø si usi 471 e non 939. Nella linea 7Ø si usi 768Ø e non 1Ø24, 384ØØ e non 55296 e, infine, N,Ø e non N,1. Nella linea 8Ø si usi 768Ø e non 1Ø24. Nelle linee 8Ø e 9Ø si usi 22 invece di 4Ø.

```
20 PRINT "♥"
30 PRINT TAB(18) "■■■■■■■
  ■■■■■■■■■■■■■
  ■■▄↑▄"
40GETK$:IFK$=""THENGOTO40
50IFK$="F"THENN=939:GOTO70
60GOTO40
70 POKE1024+N,30:POKE55296+N,1
80 POKE 1024+N+40,32
90 N=N-40
100 IF N>0 THEN GOTO 70
110 IF N<0 THEN GOTO 20
```

Per creare la base di lancio, nella parte inferiore dello schermo, viene usato un carattere ROM a bassa risoluzione (linea 3Ø). Le linee 4Ø, 5Ø e 6Ø hanno l'effetto di far attendere la pressione del tasto [F], provocando il lancio del missile quando ciò sia avvenuto assegnando a N un valore che rappresenta la posizione del missile *dopo* il lancio.

La linea 7Ø visualizza il missile, la linea 9Ø lo muove verso l'alto e la linea 8Ø cancella l'immagine dalla posizione precedente, per creare il movimento.

Le linee 1ØØ e 11Ø controllano che il missile non superi i bordi dello schermo, riavviando il programma quando ciò avviene.

MOVIMENTO SULLO SCHERMO

Nella sua forma attuale, il programma del missile non è molto divertente, ma lo si può movimentare se si permette lo spostamento trasversale della base di lancio:

```
20 CLS
30 LET P=397
40 PRINT@ P,CHR$(143)+CHR$(140)+
   CHR$(128)+CHR$(140)+CHR$(143)
50 LET K$=INKEY$:IF K$="" THEN
   GOTO 50
60 IF K$="L" THEN LET P=P-1:
   GOTO 90
70 IF K$="R" THEN LET P=P+1:GOTO90
80 GOTO 50
90 IF P<384 OR P>411 THEN GOTO 50
100 GOTO 40
```

La linea 3Ø assegna la posizione iniziale della base e la linea 4Ø visualizza il missile in tale posizione.

La linea 5Ø esamina la tastiera, in attesa di un carattere.

La linea 6Ø controlla se il carattere è una S. In caso affermativo, la base si muove di una posizione a sinistra, sottraendo 1 al valore di P.

La linea 7Ø ha la funzione di controllare se si tratta, invece, di una D per lo spostamento a destra.

La linea 9Ø ha il compito di assicurare che la base non esca dai margini dello schermo (in tal caso l'esecuzione passa alla linea 5Ø).

Si possono scegliere tasti qualsiasi per il funzionamento di un gioco?

Sì. Basta cambiare le lettere associate alle istruzioni GET$ e INKEY$. Attenzione però, perché ciò che teoricamente sembra logico, talvolta risulta negativo in pratica. Infatti, la scelta dei tasti deve dipendere, soprattutto, dalla comodità di operazione. La barra spaziatrice, in genere, serve egregiamente per 'sparare' proiettili e missili.

Questo programma consente di muovere la base di lancio sullo schermo. Sullo ZX81, usare soltanto lettere maiuscole:

```
30 CLS
40 LET x=15
50 LET y=13
60 PRINT AT y,x;"▄█▄"
70 IF INKEY$="" THEN GOTO 70
80 LET lx=x: LET ly=y
90 PRINT AT ly,lx;"□□□"
100 IF INEKY$="q" THEN STOP
110 IF INKEY$="p" THEN LET y=y-1
120 IF INKEY$="l" THEN LET y=y+1
130 IF INKEY$="z" THEN LET x=x-1
140 IF INKEY$="x" THEN LET x=x+1
150 IF x<1 OR x>29 THEN LET x=lx
160 IF y<1 OR y>20 THEN LET y=ly
170 GOTO 60
```

Le linee 4Ø e 5Ø assegnano la posizione iniziale alla base: 13 righe dall'alto e 15 colonne da sinistra. La linea 6Ø visualizza la base sullo schermo.

La linea 7Ø attende la pressione di un tasto. Le linee 8Ø e 9Ø non hanno forse un significato immediato: esse servono per cancellare la precedente posizione della base. Poiché l'esecuzione della linea 6Ø avviene sempre prima della 9Ø, la nuova posizione viene visualizzata prima di cancellare la vecchia.

La linea 1ØØ termina il programma se viene premuto il tasto [Q].

La linea 11Ø controlla se è stata digitata una p. In tal caso viene sottratto 1 dal valore di y, facendo 'salire' la base.

Le linee da 12Ø a 14Ø hanno un funzionamento analogo, ma il movimento è verso il basso con la l (linea 12Ø), verso sinistra con la z (linea 13Ø) e verso destra con

la x (linea 140).

La linea 150 evita che lo spostamento avvenga fuori dai limiti laterali dello schermo.

La linea 160 ha una funzione simile, ma per i limiti superiore ed inferiore, ponendo y = ly in caso di sconfinamento.

La linea 170, infine, completa il ciclo, rimandando l'esecuzione alla linea 60.

Questo programma muove la base di lancio attraverso lo schermo:

```
20 VDU 23;8202;0;0;0;
30 CLS
40 X=19
50 Y=13
60 PRINT TAB(X,Y)"# ∧ #"
70 KEY$=GET$
80 LX=X:LY=Y
90 PRINT TAB(LX,LY)"□□□"
100 IF KEY$="Q" THEN END
110 IF KEY$="P" THEN Y=Y-1
120 IF KEY$="L" THEN Y=Y+1
130 IF KEY$="Z" THEN X=X-1
140 IF KEY$="X" THEN X=X+1
150 IF X<1 OR X>36 THEN X=LX
160 IF Y<1 OR Y>23 THEN Y=LY
170 GOTO 60
```

Lanciato il programma, si vedrà la base al centro dello schermo. Lo spostamento si ottiene usando i tasti [Z] (sinistra), [X] (destra), [P] (alto) ed [L] (basso). Il tasto [Q] serve per uscire dal programma.

La linea 20 spegne il cursore lampeggiante. Le linee 40 e 50 assegnano alla base la posizione iniziale, mentre la 60 la visualizza. La linea 70 attende la pressione di un tasto.

Per il funzionamento delle linee 80 e 90 si veda quanto detto per il programma Spectrum (pagina precedente).

Le linee da 100 a 140 controllano quale tasto sia stato battuto e agiscono di conseguenza. Le linee 150 e 160 impediscono alla base di uscire dal campo visivo.

La linea 170 rimanda l'esecuzione alla linea 60, allo scopo di visualizzare la nuova posizione.

Sul Vic 20 si usi 8 (non 18) alla linea 20. Omettere due ■ nella linea 40 e usare 18 (e non 36) nella linea 90.

```
20 P=18
30 PRINT "■"
40 PRINT TAB(P) "■■■■■■■■
   ■■■■■■■■■■■■
   ■▀↑▀"
50 GET K$ : IF K$="" THEN GOTO 50
60 IF K$ "L" THEN P=P-1 : GOTO 90
70 IF K$="R" THEN P=P+1 : GOTO 90
80 GOTO 50
90 IF P>36 THEN P=36
100 IF P<1 THEN P=1
110 GOTO 30
```

La linea 20 assegna la posizione iniziale della base a P e la linea 40 visualizza la base in tale posizione. Il controllo su quale tasto venga premuto avviene alle linee 50, 60 e 70: S sposta la base a sinistra, D la sposta a destra, modificando il valore di P. La GOTO nella linea 80 fa tornare il programma alla linea 50 se viene battuto qualsiasi altro tasto.

Le linee 90 e 100 controllano che la base non esca dai limiti consentiti, ripetendo la GET se ciò accade. Infine, la linea 110 rinvia il programma alla linea che visualizza la base.

LA CREAZIONE DI UN GIOCO

Adesso possediamo vari elementi coi quali costruire giochi. Qui sotto riportiamo un esempio del loro impiego. Sul Tandy si usi 247 (e non 223) alle linee 100 e 110.

```
20 CLS
30 FOR N=1 TO 100:NEXT N
40 LET PO=430
50 LET B$=CHR$(143)+CHR$(140)+
   CHR$(128)+CHR$(140)+CHR$(143)
60 LET A=RND(30)+64
70 PRINT@ A,"*"
80 LET LP=PO
90 PRINT@ PO,B$
100 IF PEEK(340)=223 THEN LET PO=PO
   -1
110 IF PEEK(338)=223 THEN LET PO=PO
   +1
120 IF PO<415 OR PO>444 THEN LET PO
   =LP
130 LET K$=INKEY$
140 IF K$="F" THEN LET M=PO-
   30 ELSE GOTO 80
150 PRINT@ M,"↑";
160 PRINT@ M+32,"□";
170 LET M=M-32
180 IF M=A THEN GOTO 20
190 M>0 THEN GOTO 150 ELSE PRINT
   @ M+32,"□";
200 GOTO 80
```

Sullo ZX81 usare soltanto lettere maiuscole e sostituire la freccetta con un asterisco:

```
20 CLS
30 PAUSE 25
40 LET x=15: LET y=21
50 LET BS="□▄■▄□"
60 LET a=INT (RND*28)+2
70 PRINT AT 2,a;"*"
80 LET xx=x
90 PRINT AT y,x;B$
100 IF INKEY$="z" THEN LET x=x-1
110 IF INKEY$="x" THEN LET x=x+1
120 IF x<0 OR x>27 THEN LET x=xx
140 IF INKEY$<>"f" THEN GOTO 80
145 LET m=y-1
150 PRINT AT m,x+2;"↑"
160 PRINT AT m+1,x+2;"□"
170 LET m=m-1
180 IF m=2 AND x+2=a THEN GOTO 20
190 IF m<>1 THEN GOTO 150
195 PRINT AT m+1,x+2;"□"
200 GOTO 80
```

```
15 VDU 23;8202;0;0;0;
20 CLS
30 FOR N=1 TO 200:NEXT N
40 LET X=19: LET Y=20
50 LET B$="□ # ∧ # □"
60 LET AX=RND(36)+1
70 PRINT TAB(AX,3) "*"
80 LET LX=X
90 PRINT TAB(X,Y); B$
95 LET K$=GET$
100 IF K$="Z" THEN LET X=X-1
110 IF K$="X" THEN LET X=X+1
120 IF X<0 OR X>35 THEN LET X=LX
140 IFK$="F" THEN LET M=
   19 ELSE GOTO 80
150 PRINT TAB(LX+2,M) "∧"
160 PRINT TAB(LX+2,M+1) "□"
170 LET M=M-1
180 IF M=3 AND LX+2=
   AX THEN GOTO 20
190 IF M>0 THEN GOTO 150 ELSE PRINT
```

D+R

Perché un programma può bloccarsi se un simbolo grafico giunge al margine dello schermo?

Probabilmente, uno dei numeri che controllano la posizione del simbolo sullo schermo è troppo grande. Si ricontrollino le linee di programma che modificano tali valori, se si ottengono messaggi del tipo: 'screen scrolling' (Acorn), 'illegal quantity' (Commodore), 'F/C error' (Dragon), 'integer out of range' (Spectrum).

```
   TAB(LX+2,M+1) "□"
200 GOTO 80
```

Sul Vic 2Ø, sostituire la linea 15 con 15 POKE 36879,29. Cambiare, nella linea 4Ø, il 16 in 8. Alle linee 6Ø e 1Ø5 cambiare i 34 in 16. Omettere due ▣ nella linea 5Ø.

```
15 POKE 53280,5:POKE 53281,1
20 PRINT "▢"
30 CLR
40 LET P=16:LET A=1
50 LET D$="▣▣▣▣▣▣▣▣▣
   ▣▣▣▣▣▣▣▣▣▣
   ▣"
60 LET A=INT(RND(1)*34)+3
70 PRINT "▤"TAB(A)"♣"
80 PRINT "▤" TAB(P)D$"□▄ ↑ ▄□"
90 GET K$:IF K$="" THEN 90
95 IF K$="Z" THEN P=P-1
100 IF K$="X" THEN P=P+1
105 IF P>34 THEN P=34
110 IF P<1 THEN P=1
115 IF K$="F" THEN P1=P:D=22:GOTO
   130
120 GOTO 80
130 PRINT "▤" TAB(P1);
140 PRINT LEFT$(D$,D)"▌▌ ↑ ▣▌
   □":D=D-1
150 PRINT "▤" TAB(P1):
160 PRINT LEFT$(D$,D)"▌▌□▣▌□":D
   =D-1
170 IF D>0 THEN 130
180 IF P1=A-2 THEN 20
200 GOTO 80
```

Lanciando il programma, si vedrà una stella in alto sullo schermo. Il giocatore deve distruggerla usando i tasti Z e X per muovere la base di lancio e F per far fuoco. Se si riesce nell'intento, il gioco ricomincia.

Si pensi al programma come composto da tre sezioni: dalla linea 15 alla 7Ø, dalla 8Ø alla 12Ø e dalla 13Ø/14Ø alla 2ØØ.

Le linee da 13Ø/14Ø a 2ØØ provvedono al lancio del missile e assomigliano a quanto presentato precedentemente. I nomi delle variabili e le GOTO sono diverse, ma l'unica aggiunta è la linea 18Ø, che controlla se il missile e la stella occupano la medesima posizione (gioco finito).

La sezione centrale (linee da 8Ø a 12Ø) è una versione ridotta del programma di movimento attraverso lo schermo. Le PEEK nel programma per Dragon e Tandy controllano l'avvenuta pressione di una Z o di una X, modificando pertanto PO.

La prima sezione del programma (fino alla linea 7Ø) svolge varie funzioni. Nella versione Acorn, la linea 15 spegne il cursore lampeggiante. In tutti i programmi, la linea 3Ø introduce una breve pausa, importante quando la 18Ø completa il ciclo. Le linee 4Ø e 5Ø stabiliscono la posizione iniziale e la forma della base di lancio (sul Commodore la forma è definita alla linea 8Ø). Le linee 6Ø e 7Ø scelgono una posizione per la stella e la visualizzano.

UN MIGLIOR MOVIMENTO

Il fatto di dovere premere i due tasti per lo spostamento a destra e a sinistra per ottenere uno spostamento, come avviene per Dragon, Tandy e Commodore, è poco efficiente. Creiamo dunque un meccanismo di ripetizione automatica (autorepeat).

Sul Commodore ciò si ottiene semplicemente usando una POKE. Si aggiunga:

```
10 POKE 650, 128
```

(In effetti si può usare un qualsiasi valore superiore a 128). Per annullare l'autorepeat, si esegua:

```
POKE 650, 127.
```

Un movimento scorrevole, con l'impiego della INKEY$ non è facile da ottenere. Si può però aggirare il problema come illustrato dal seguente programma. Sul Tandy si usi 247 (e non 223) nelle linee 7Ø e 8Ø; si usi 251 (e non 239) nella linea 9Ø; si usi 253 (e non 247) nella linea 1ØØ.

```
20 CLS
30 LET BL$=CHR$(128)
40 LET PO=238
50 PRINT@ PO,BLS
60 LET LP=PO
70 IF PEEK(340)=223 THEN LET PO=PO
   -1:GOTO 120
80 IF PEEK(338)=223 THEN LET PO=PO
   +1:GOTO 140
90 IF PEEK(338)=239 THEN LET PO=PO
   -32:GOTO 150
100 IF PEEK(342)=247 THEN LET PO=PO
   +32:GOTO 150
110 GOTO 70
120 IF (LP AND 31)=0 THEN LET PO=LP
130 GOTO 150
140 IF (PO AND 31)=0 THEN LET PO=LP
150 IF PO>510 OR PO<0 THEN LET
   PO=LP:GOTO 70
160 PRINT@ LP, "□";
170 PRINT@ PO,BL$;
180 GOTO 60
```

Una volta lanciato il programma con un RUN, si potrà muovere il quadratino che appare sullo schermo in tutte le direzioni.

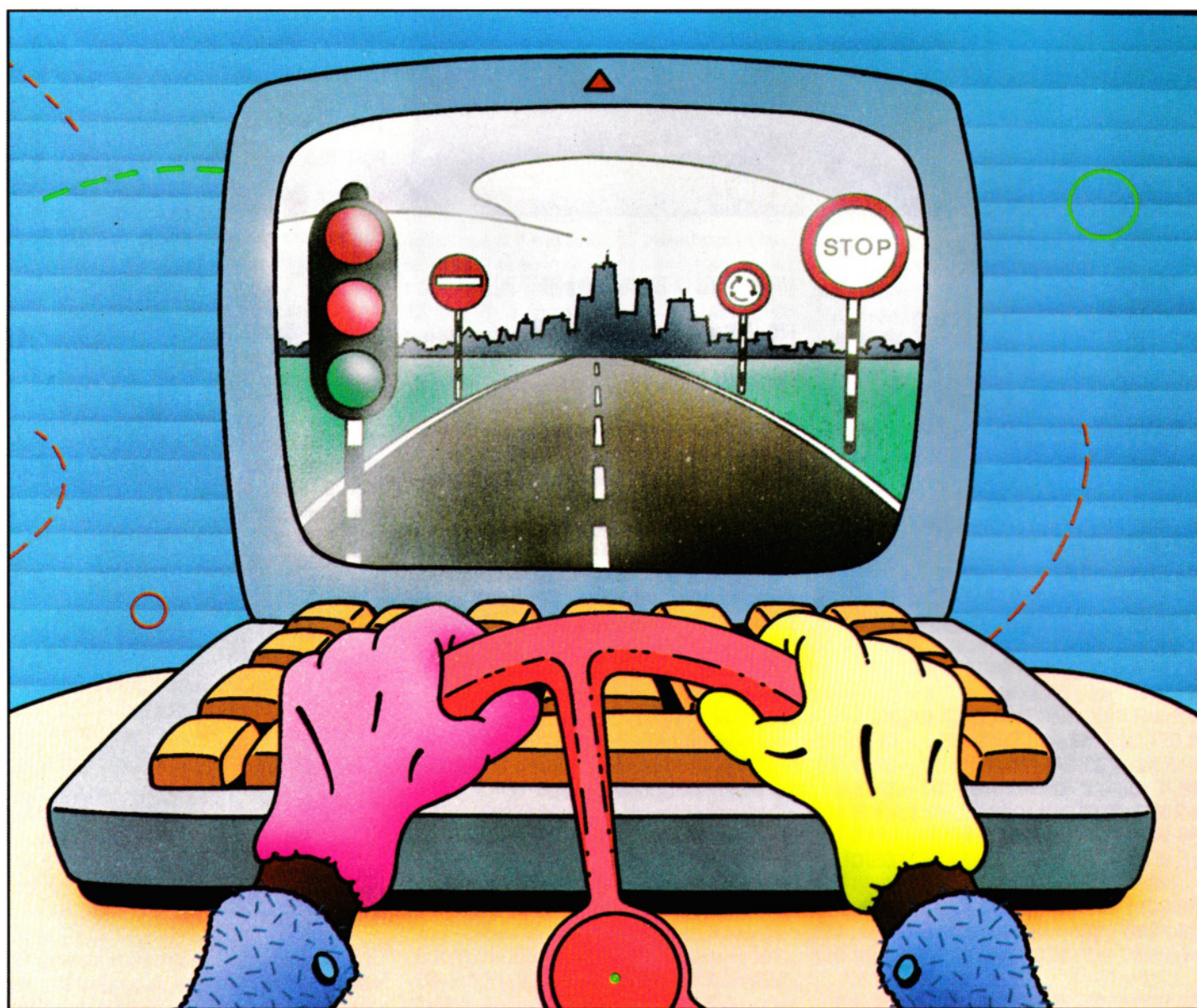
I SEGNALI STRADALI DEL PROGRAMMATORE

Un attento esame di due istruzioni, la GOTO e la GOSUB, rivela che esse sono molto utili per pilotare l'esecuzione dei programmi

Una delle istruzioni fondamentali della programmazione in BASIC è la GOTO. La sua funzione è di modificare la sequenza d'esecuzione di un programma in modo da *saltare* alla linea indicata dopo la GOTO, anziché eseguire le linee sequenzialmente, cioè una dopo l'altra.

Benché sullo schermo, talvolta, appaia col formato GO TO, l'istruzione è composta da una sola parola. Sullo Spectrum, l'immissione è facilitata dalla presenza di un apposito tasto [GOTO], su altri computer occorre digitare la parola per intero.

La GOTO è sempre seguita dal numero della linea alla quale deve saltare. Su alcuni computer (non sul Commodore, però), questo numero si può sostituire con una variabile (A, per esempio), alla quale viene assegnato un valore in altra parte del programma.

Se usata per saltare all'indietro nel programma, la GOTO serve a creare dei cicli, simili a quelli ottenuti con FOR ... NEXT (si veda a pagina 16), ma senza un limite al numero di iterazioni. Il seguente programma, per esempio, calcola la lunghezza dell'ipotenusa di un triangolo retto. (Si noti come i valori di A e B vengano elevati al quadrato senza ricorrere al formato A ↑ 2 e B ↑ 2).

- LE **GOTO** E **GOSUB** IN PRATICA - PROGRAMMI DI CALCOLO, INDOVINELLI E LANCIO DI DADI
- COME E QUANDO CREARE I SALTI IN AVANTI E ALL'INDIETRO NEI PROGRAMMI
- LE DIRAMAZIONI COMPLESSE
- SVELTIRE LE SUBROUTINE
- USO DELLE **PROCEDURE**
- COME EVITARE UNA CATTIVA PROGRAMMAZIONE DOVUTA ALL'USO DI **GOTO**

```
10 PRINT "Lunghezza dei lati A e B (cm)"
15 INPUT A,B
20 LET C=SQR (A*A+B*B)
30 PRINT "Il lato C misura □"; C;
   "□centimetri."
40 GOTO 10
```

```
10 PRINT '"Lunghezza dei lati A e B (cm)"
15 INPUT a,b
20 LET c=SQR (a*a+b*b)
30 PRINT "Il lato C misura □"'c;
   "□centimetri."
40 GOTO 10
```

```
10 PRINT "LUNGHEZZA DEI LATI A E B
   (CM)"
15 INPUT A,B
20 LET C=SQR(A*A+B*B)
30 PRINT "IL LATO C MISURA"; C;"CM"
40 GOTO 10
```

Alla linea 2Ø, SQR (A*A+B*B) significa 'la radice quadrata di A al quadrato più B al quadrato', ossia: $\sqrt{A^2+B^2}$, in base al teorema di Pitagora. La linea 3Ø visualizza il risultato.

Il programma si ripete all'infinito, poiché ogni volta che l'esecuzione raggiunge la linea 4Ø, la GOTO fa ripartire il programma dalla linea 1Ø. L'unico modo per uscire dal ciclo consiste nel premere ESCAPE, BREAK oppure STOP/RUN (sullo Spectrum battere STOP durante la INPUT). Oppure spegnere il computer.

UN SALTO IN AVANTI

La GOTO si può usare anche per saltare oltre un gruppo di istruzioni, come mostrato in questo programma per il lancio di una moneta:

```
5 ritardo=INKEY(200):CLS
10 PRINT "STO LANCIANDO LA MONETA...";
20 FOR J=1 TO 3
30 PRINT ".";
40 ritardo=INKEY(100)
50 NEXT
70 IF RND(1)<0,5 THEN GOTO 100
80 PRINT' "È CROCE!"
90 GOTO 5
100 PRINT' "È TESTA!"
110 GOTO 5
```

```
5 PAUSE 50: CLS
10 PRINT "STO LANCIANDO
   LA MONETA...";
20 FOR j=1 TO 3
30 PAUSE 25
40 PRINT ".";
50 NEXT j
60 PRINT
70 IF RND<.5 THEN GOTO 100
80 PRINT "È CROCE!"
90 GOTO 5
100 PRINT "È TESTA"
110 GOTO 5
```

```
5 FOR F=1 TO 500:NEXT F:CLS
10 PRINT "STO LANCIANDO LA MONETA...";
20 FOR J=1 TO 3
30 FOR F=1 TO 250:NEXT F
40 PRINT ".";
50 NEXT J
60 PRINT
70 IF RND(0)<.5 THEN GOTO 100
80 PRINT "È CROCE!"
90 GOTO 5
```

```
100 PRINT "È TESTA!"
110 GOTO 5
```

```
5 FOR D=1 TO 1000 : NEXT : PRINT "♥"
10 PRINT "STO LANCIANDO LA MONETA...";
20 FOR J=1 TO 3
30 FOR D=1 TO 250 : NEXT D
40 PRINT ".";
50 NEXT J
60 PRINT
70 IF RND(0)<.5 THEN GOTO 100
80 PRINT "È CROCE!"
90 GOTO 5
100 PRINT "È TESTA!"
110 GOTO 5
```

La RND alla linea 70 genera un numero a caso, compreso tra 0 e 1.

In questo caso, l'istruzione fa parte di una frase condizionale, che termina con una GOTO.

Se il numero generato dal computer è minore di 0,5, l'esecuzione passa alla linea successiva, mentre viene ignorato il resto dalla linea 70.

Ciò significa che la linea 70 forma una diramazione nel programma: o sono eseguite le linee 70, 80 e 90 (visualizzando il messaggio "CROCE!") oppure le linee 70, 100 e 110, (visualizzando "TESTA!"): il computer lancia una moneta elettronica.

Anche le linee 90 e 110 contengono delle GOTO: qualunque diramazione abbia preso il computer, queste rimandano sempre all'inizio del programma (linea 5) e tutto ricomincia.

Di nuovo, per uscire dal ciclo senza fine, occorre premere ESCAPE, BREAK o RUN/STOP o spegnere il computer.

DIRAMAZIONI PIU' COMPLESSE

Sui computer Acorn e Spectrum, il numero che segue una GOTO si può anche sostituire con una variabile numerica o con un'espressione: GOTO A, oppure GOTO (100 + INT (RND*6)) sono formati validi.

Ecco di seguito un esempio di diramazione complessa:

```
100 PRINT "Ciao, come ti chiami?"
110 INPUT A$
120 GOTO (120+RND(4)*10)
130 PRINT "Che bel nome,□";A$: GOTO 170
140 PRINT "Che nome buffo,□";A$ GOTO
   170
150 PRINT "Benvenuto,□";A$:GOTO 170
160 PRINT "Ciao,□";A$;"□sono il tuo
   computer"
170 END
```

```
100 PRINT "Ciao, come ti chiami?"
110 INPUT a$
120 GOTO (130+INT (RND*4)*10)
130 PRINT "Che bel nome,□";a$:GOTO 170
140 PRINT "Che nome buffo,□";a$:GOTO 170
150 PRINT "Benvenuto,□";a$: GOTO 170
160 PRINT "Ciao,□";a$;",□sono il tuo
   computer."
170 STOP
```

Nella linea 120, la linea alla quale salta la GOTO è calcolata, a caso, tra le quattro che seguono. Un simile sistema è utile nei giochi, quando nel programma si vuol creare un percorso a sorpresa e del tutto imprevedibile.

Le varie GOTO 170 provocano un salto in avanti, evitando l'esecuzione delle linee intermedie. Si noti l'inutilità di aggiungere una GOTO 170 al termine della linea 160, in quanto da essa si passa in ogni caso alla 170.

Il programma *gira* una sola volta, poiché le GOTO saltano tutte in avanti, senza mai creare un ciclo. La linea 170 termina l'esecuzione del programma.

ON ... GOTO

Sul Commodore, sul Dragon e sugli Acorn esiste la struttura ON ... GOTO, simile a quella appena vista. Ecco il formato:

```
ON A GOTO 100, 200, 300, 400
```

Se A=1, l'esecuzione salta alla prima linea specificata (la 100), se A=2 il salto è alla seconda linea (la 200), e così via.

Anche in questo caso, si possono creare diramazioni complesse.

Riscriviamo l'esempio precedente:

```
100 PRINT "CIAO, COME TI CHIAMI?"
110 INPUT A$
120 ON RND(4) GOTO 130,140,150,160
130 PRINT "CHE BEL NOME,□";A$:GOTO 170
140 PRINT "CHE NOME BUFFO,□";A$:GOTO
   170
150 PRINT "BENVENUTO,□";A$:GOTO 170
160 PRINT "CIAO,□";A$;",□SONO IL TUO
   COMPUTER"
170 END
```

```
100 PRINT "CIAO, COME TI CHIAMI?"
110 INPUT A$
120 ONINT(RND(1)*5)GOTO 130,140,150,160
130 PRINT "CHE BEL NOME,□";A$:GOTO 170
140 PRINT "CHE NOME BUFFO,□";A$:GOTO
   170
150 PRINT "BENVENUTO,□";A$:GOTO 170
160 PRINT "CIAO,□"A$ "□SONO IL TUO
   COMPUTER"
170 END
```

BUONA E CATTIVA PROGRAMMAZIONE

Abusare delle GOTO è considerato un pessimo stile di programmazione. Anche nei programmi più semplici, infatti, è facile creare inavvertitamente cicli senza fine, dai quali si esce soltanto battendo ESCAPE, BREAK o STOP/RUN, oppure spegnendo il computer.

Ma la ragione principale è che, permettendo di saltare a piacimento in avanti e all'indietro, le GOTO frammentano la struttura logica del programma. Ciò può sembrare irrelevante in programmi lunghi una decina di linee, ma in quelli di maggiori dimensioni l'effetto può essere catastrofico.

Un buono stile di programmazione si ottiene soltanto suddividendo i programmi in moduli logici, ciascuno con una precisa funzione. Ciò facilita l'individuazione di eventuali errori e la comprensione dell'intero programma. Inoltre, i moduli si possono riutilizzare in altri programmi, senza doverli riscrivere da capo.

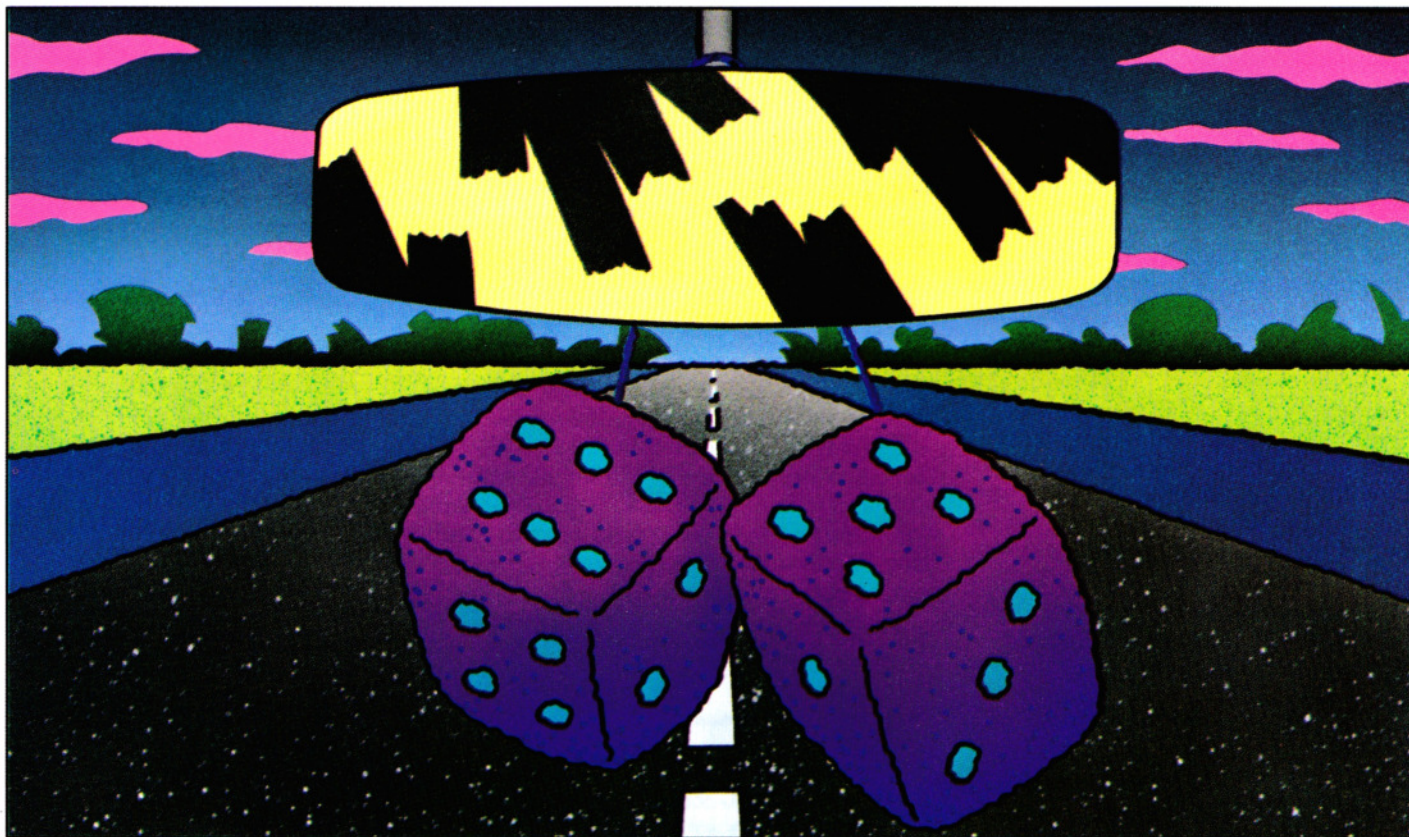
USO DELLE GOSUB

Nella 'borsa degli attrezzi' del programmatore ce n'è uno che rimpiazza egregiamente la GOTO. Si tratta della GOSUB. Anche questa è sempre seguita da un numero di linea.

La GOSUB devia il normale corso dell'esecuzione, consentendo di eseguire un sottoprogramma o *subroutine*, che inizia dal numero di linea specificato. Una subroutine non è che una serie di istruzioni formanti un blocco logico indipendente ed è spesso usata quando una stessa operazione deve essere eseguita più volte in un programma.

La fondamentale differenza tra la GOSUB e la GOTO è che la subroutine *chiamata* dalla GOSUB deve terminare con una RETURN.

Attenzione a non confondere l'*istruzione* RETURN col *tasto* RETURN! Sullo Spectrum, l'istruzione RETURN si immette con l'apposito tasto, ma negli altri computer occorre digitare per intero la parola, prima di battere il tasto RETURN o ENTER. L'istruzione RETURN fa riprendere l'esecuzione dalla linea successiva a quella che contiene la relativa GOSUB o, come nello Spectrum, all'istruzione seguente alla GOSUB.

Il seguente programma simula un gioco di dadi americano, il *Craps*.

I dadi vengono lanciati per due volte, annotando i punti di ciascun lancio. Se due lanci producono lo stesso totale, il gioco finisce.

```
20 LET A=1
30 REM. .PRIMO LANCIO. .
40 GOSUB 150
50 LET T1=T
60 REM. .SECONDO LANCIO. .
70 GOSUB 150
80 LET T2=T
90 IF T1=T2 THEN GOTO 120
100 LET A=A+1
110 GOTO 40
120 PRINT "TOTALI IDENTICI DI□";T1;
    "□IN□";A,"□LANCI"
130 END
140 REM. .SUBROUTINE. .
150 LET D1=RND(6)
160 LET D2=RND(6)
170 LET T=D1+D2
180 RETURN
```

```
20 LET A=1
30 REM ***PRIMO LANCIO**
40 GOSUB 150
50 LET T1=T
60 REM ***SECONDO LANCIO***
70 GOSUB 150
80 LET T2=T
90 IF T1=T2 THEN GOTO 120
100 LET A=A+1
110 GOTO 40
120 PRINT "TOTALI IDENTICI DI" T1 "IN" A
    "LANCI"
130 END
140 REM ***SUBROUTINE***
150 LET D1=INT(RND(X)*6+1)
160 LET D2=INT(RND(X)*6+1)
170 LET T=D1+D2
180 RETURN
```

Sullo ZX81, usare soltanto lettere maiuscole:

```
20 LET a=1
30 REM primo lancio
40 GOSUB 150
50 LET t1=T
60 REM secondo lancio
70 GOSUB 150
80 LET t2=T
90 IF t1=t2 THEN GOTO 120
100 LET a=a+1
110 GOTO 40
120 PRINT "Totali identici di□";t1;
    "□in□";a;"□lanci"
130 GOTO 200
140 REM subroutine
150 LET d1=INT (RND*6)+1
160 LET d2=INT (RDN*6)+1
170 LET T=d1+d2
180 RETURN
```

Poiché il lancio dei dadi avviene due volte, viene usata una subroutine, costituita dalle linee da 150 a 180.

Le GOSUB nelle linee 40 e 70 chiamano la subroutine.

La RETURN nella linea 180 rispedisce l'esecuzione, rispettivamente, alla linea 50 e alla linea 80. La linea 140 serve unicamente da commento e non fa parte della subroutine.

Non conviene includere le frasi di commento REM nelle subroutine, poiché queste linee verrebbero ripetute dal computer senza alcun effetto pratico, se non quello di allungare i tempi.

È preferibile assegnare loro un numero di linea immediatamente precedente a quello della subroutine.

Si noti la END alla linea 130 (eccetto nel programma per i Sinclair).

Qualora venisse omessa, il computer, giunto alla linea 120, passerebbe a eseguire la subroutine, visualizzando un messaggio d'errore quando incontra la RETURN (linea 180) senza che vi sia stata una GOSUB.

UNA GOTO FUORI CAMPO

Poiché lo Spectrum non dispone della END, al suo posto si è usata una GOTO, seguita da un numero di linea superiore all'ultimo usato nel programma.

Abbiamo usato, infatti:

```
130 GOTO 200
```

Non trovando nessuna linea uguale o maggiore di 200, lo Spectrum considera terminato il programma, visualizzando il normale OK. Questo sistema è migliore di una semplice:

```
130 STOP
```

perché la STOP, nello Spectrum, provoca la visualizzazione di un avviso (nel nostro caso: 9 STOP statement 130:1) e ciò può far credere di aver commesso qualche errore, anche perché la linea 130 non è l'ultima del programma, ma precede, invece, la subroutine.

Occorre però assicurarsi che non esistano, effettivamente, delle linee di programma successive a quella indicata nella GOTO. Per andare sul sicuro, conviene far saltare l'esecuzione all'ultimo numero di linea possibile, il 9999.

Il metodo più corretto per i Sinclair è di usare GOTO 9999 e di aggiungere in coda al programma:

```
9999 REM END
```

L'AGGIUNTA DI PIU' LIVELLI

Una subroutine può, a sua volta, chiamare un'altra subroutine (anche se stessa!), creando così più di un livello d'esecuzione. Siccome il lancio di dadi è ripetuto due volte, possiamo costruire una subroutine interna a quella originale, ottenendo:

```
150 GOSUB 190
155 LET D1=D
160 GOSUB 190
165 LET D2=D
170 LET T=D1+D2
180 RETURN
190 LET D=RND(6)
195 RETURN
```

```
150 GOSUB 190
155 LET D1=D
160 GOSUB 190
165 LET D2=D
170 LET T=D1+D2
180 RETURN
190 LET D=INT(RND(X)*6+1)
200 RETURN
```

Sullo ZX81, usare soltanto lettere maiuscole:

```
150 GOSUB 190
155 LET d1=d
160 GOSUB 190
165 LET d2=d
170 LET T=d1+d2
180 RETURN
190 LET d=INT (RND*6)+1
195 RETURN
```

Qui, le linee 150 e 160 servono per chiamare la subroutine alla linea 190 (singolo lancio del dado), mentre le linee 155 e 165 annotano separatamente i risultati dei due lanci.

D'altra parte, eccezion fatta per i Sinclair, la subroutine può essere più efficacemente riscritta così:

```
150 LET T=INT(6*RND(1)+1)+INT(6*
  RND(1)+1)
```

Oppure, così:

```
150 T=INT(6*RND+1)+INT(6*RND+1)
```

In alcuni computer va anche bene:

```
150 T=RND (6)+RND (6)
```

Un abile programmatore, sul Dragon sul Tandy o sugli Acorn, riesce a riunire l'intero programma in due sole righe:

```
10 A=0
20 D1=RND(6)+RND(6):D2=RND(6)+6:A
  =A+1:IF D1=
  D2 THEN PRINT "TOTALI IDENTICI DI□";
  D1;"□IN□";A;"LANCI"
  : END ELSE GOTO 20
```

USO DI ON ... GOSUB

Su alcuni computer, anche con la GOSUB, come per la GOTO, si può usare una variabile al posto di un numero di linea. Altri, invece, consentono l'uso di ON ... GOSUB. Il funzionamento è analogo a quello visto per ON ... GOTO, salvo che, in questo caso, vengono eseguite delle subroutine.

LE PROCEDURE NEL MICRO BBC

Il BASIC del BBC consente l'uso di procedure. Queste sono simili a delle subroutine, costituendo sezioni separate di programma, richiamate dal programma principale. Diversamente dalla subroutine, le procedure, molto più versatili, possono essere chiamate *per nome*. Eccone un esempio:

```
10 CLS
20 FOR mese=1 TO 12
30 PRINT TAB(0,1)"Valori per il
  mese□";mese
40 INPUT N
50 PROCdrawgraph
60 PRINT TAB(0,2)"□□□"
70 NEXT mese:END
90 DEFPROCdrwgraph
100 FOR X=1 TO N
110 PRINT TAB(X,mese+6)"*";
120 NEXT
130 ENDPROC
```

Questo programma traccia un grafico in base a 12 valori, uno per mese. I valori potrebbero rappresentare il consumo mensile di benzina per l'auto, o qualsiasi altro ammontare variabile. (Nell'esempio, il massimo valore è 39, per rientrare nello schermo).

La procedura (PROCgrafico) viene chiamata 12 volte, una per mese. Si noti che PROCgrafico *chiama* la procedura, mentre DEFPROCgrafico la *definisce*, e ENDPROC rimanda al programma principale. A una procedura si possono anche passare dei valori (chiamati *parametri*), cosicché una singola procedura può essere usata in circostanze diverse tra loro.

Il seguente programma usa una procedura generica per disegnare sullo schermo, in varie posizioni, tre linee colorate.

```
10 INPUT "COME TI CHIAMI",N$
15 IF LEN(N$)>20 THEN GOTO 10
20 MODE 2
30 VDU 23;8202;0;0;0;
40 PROCdisplay(5,1,130,"BUON NATALE")
50 PROCdisplay(12,11,140,N$)
60 PROCdisplay(20,1,
130,"E FELICE ANNO NUOVO")
70 END
100 DEFPROCdisplay(riga,colore,carta,
  messaggio$)
105 LOCALX,L
110 LET L=LEN(messaggio$)
120 col=INT((20-L)/2)
130 COLOUR colore
140 COLOUR carta
150 PRINT TAB(col,riga);messaggio$
160 FOR X=0 TO L-1
170 PRINT TAB(col+X,riga-1) "*"
180 PRINT TAB(col+X,riga+1) "*";
190 NEXT
200 ENDPROC
```

Qualsiasi tipo di testo può esser visualizzato con questa procedura, semplicemente usando parametri diversi.

INDICE CUMULATIVO DEI FASCICOLI

NEL PROSSIMO NUMERO

☐ *I LABIRINTI hanno da sempre affascinato persone di tutte le età: come crearli ed usarli sul computer e produrre nuovi giochi.*

☐ *Dedichiamoci all'arte grafica su computer, grazie ai comandi DRAW e PAINT.*

☐ *Impariamo a usare efficacemente il nostro PROGRAMMA DI ARCHIVIAZIONE aggiungendovi funzioni di ricerca e di correzione dei dati.*

☐ *L'ASSEMBLY è un linguaggio a basso livello, ma molto veloce e preciso: addentriamoci nei meandri del CODICE MACCHINA.*

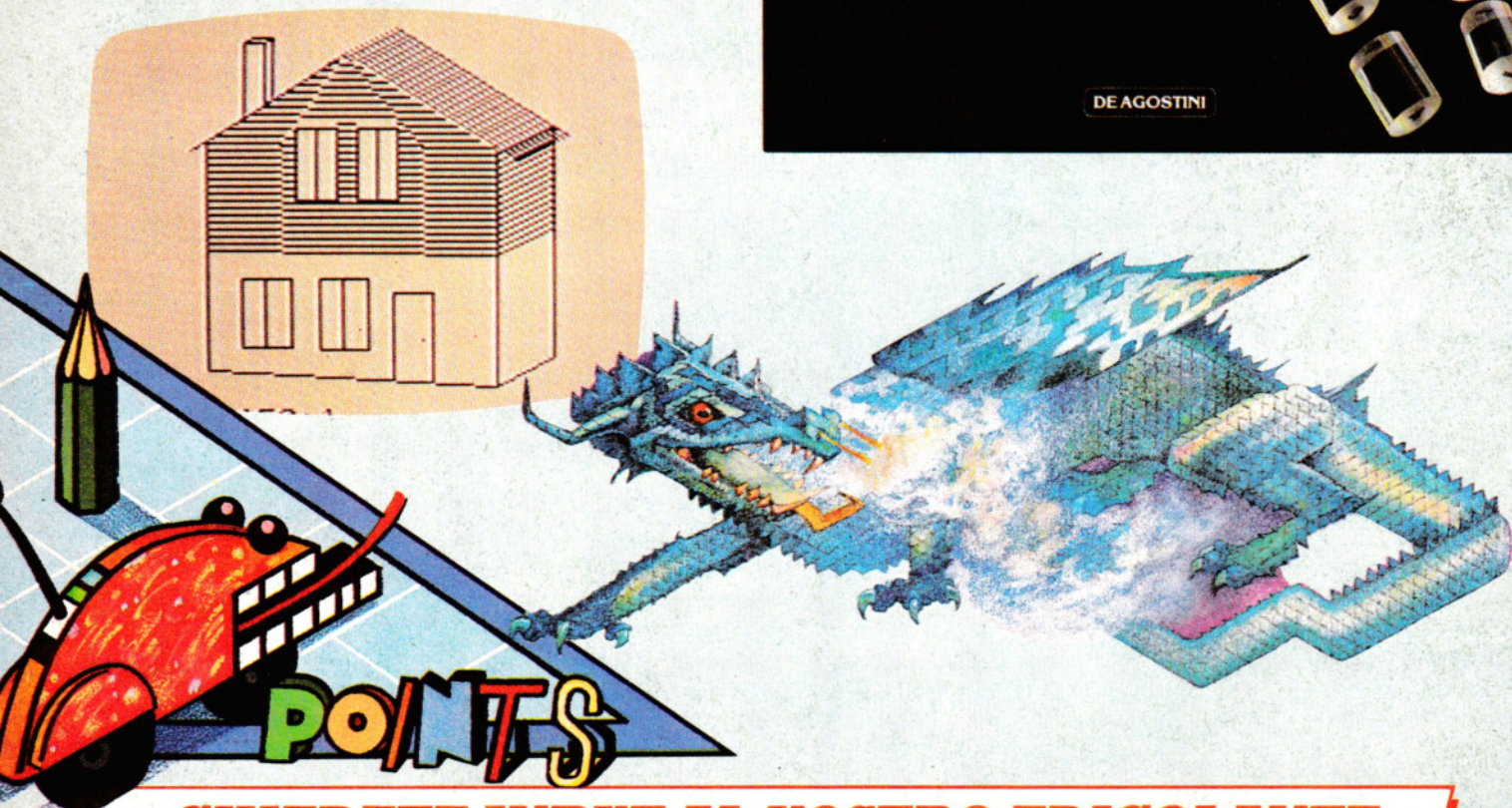
☐ *Il repertorio dei personaggi per i giochi si arricchisce: creiamo un DRAGO SPUTAFUOCO, con poche linee di programma.*

☐ *Il concetto di 'variabile', nel linguaggio BASIC, può confondere i principianti: Vediamo cosa sono le VARIABILI e a cosa servono nei programmi.*

INPUT 3

CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER

DE AGOSTINI

POINTS

CHIEDETE INPUT AL VOSTRO EDICOLANTE